教材

书名：《人工神经网络导论》

出版社：高等教育出版社

出版日期：2001年8月

重印：2003年2月

定价：12.4元

作者：蒋宗礼

主要参考书目

1、Philip D. Wasserman， Neural Computing: Theory and Practice，Van Nostrand Reinhold，1989

2、胡守仁、余少波、戴葵，神经网络导论，国防科技大学出版社，1993年10月

3、杨行峻、郑君里，人工神经网络，高等教育出版社，1992年9月

4、闻新、周露、王丹力、熊晓英，MATLAB神经网络应用设计，科学出版社，2001.5.

课程目的和基本要求

- 作为人工神经网络的入门课程，用于将学生引入人工神经网络及其应用的研究领域。
- 介绍人工神经网络及其基本网络模型，使学生
  - 了解智能系统描述的基本模型
  - 掌握人工神经网络的基本概念、单层网、多层网、循环网等各种基本网络模型的结构、特点、典型训练算法、运行方式、典型问题
  - 掌握软件实现方法。

课程目的和基本要求

- 了解人工神经网络的有关研究思想，从中学习开拓者们的部分问题求解方法。
- 通过实验进一步体会有关模型的用法和性能，获取一些初步的经验。
- 查阅适当的参考文献，将所学的知识与自己未来研究课题（包括研究生论文阶段的研究课题）相结合起来，达到既丰富学习内容，又有一定的研究和应用的目的。

主要内容

- 智能及其实现
- ANN基础
- Perceptron
- BP
- CPN
- 统计方法
- Hopfield网与BAM
- ART

主要内容

第一章：引论

智能的概念、智能系统的特点及其描述基本模型，物理符号系统与连接主义的观点及其比较；人工神经网络的特点、发展历史。

主要内容

第二章 人工神经网络基础

本章在介绍了基本神经元后，将概要介绍人工神经网络的一般特性。主要包括，生物神经网络模型，人工神经元模型与典型的激励函数；人工神经网络的基本拓扑特性，存储类型（**CAM——LTM**，**AM——STM**）及映象，**Supervised**训练与**Unsupervised**训练。

主要内容

第三章 感知器

感知器与人工神经网络的早期发展；单层网能解决线性可分问题，而无法解决线形不可分问题，要想解决这一问题，必须引入多层网；**Hebb**学习律，**Delta**规则，感知器的训练算法。

- 实验：实现一个感知器。

主要内容

第四章 向后传播

- BP（Backpropagation）网络的构成及其训练过程；隐藏层权调整方法的直观分析，BP训练算法中使用的Delta规则（最速下降法）的理论推导；算法的收敛速度及其改进讨论；BP网络中的几个重要问题。
- 实验：实现BP算法。

主要内容

第五章 对传网

- 生物神经系统与异构网的引入；对传网的网络结构，**Kohonen**层与**Grossberg**层的正常运行，对传网的输入向量的预处理，**Kohonen**层的训练算法及其权矩阵的初始化方法；**Grossberg**层的训练；完整的对传网。
- 实验：实现基本的对传网。

主要内容

第六章 统计方法

- 统计方法是为了解决局部极小点问题而引入的，统计网络的基本训练算法，模拟退火算法与收敛分析，**Cauchy**训练，人工热处理与临界温度在训练中的使用，**BP**算法与**Cauchy**训练相结合。
- 实验：实现模拟退火算法。

主要内容

第七章 循环网络

- 循环网络的组织，稳定性分析；相联存储；统计**Hopfield**网与**Boltzmann**机；**Hopfield**网用于解决**TSP**问题。
- BAM(Bidirectional Associative Memory)用于实现双联存储；基本双联存储网络的结构及训练；其他的几种相联存储网络。
- 实验：实现一个**Hopfield**网。

主要内容

第八章 自适应共振理论

- 人脑的稳定性与可塑性问题；**ART**模型的总体结构与分块描述；比较层与识别层之间的两个联接矩阵的初始化，识别过程与比较过程，查找的实现；训练讨论。

第1章　引言

- 主要内容：
  - 智能与人工智能；
  - ANN的特点；
  - 历史回顾与展望
- 重点：
  - 智能的本质；
  - ANN是一个非线性大规模并行处理系统
- 难点：对智能的刻画

第1章　引言

1.1 人工神经网络的提出

1.2 人工神经网络的特点

1.3 历史回顾

第1章　引言

- 人类对人工智能的研究可以分成两种方式对应着两种不同的技术：
  - 传统的人工智能技术——心理的角度模拟
  - 基于人工神经网络的技术——生理的角度模拟

1.1 人工神经网络的提出

- 人工神经网络（**Artificial Neural Networks**，简记作**ANN**），是对人类大脑系统的一阶特性的一种描述。简单地讲，它是一个数学模型，可以用电子线路来实现，也可以用计算机程序来模拟，是人工智能研究的一种方法。

1.1 人工神经网络的提出

- **1.1.1 智能与人工智能**
- 一、 智能的含义
- 智能是个体有目的的行为，合理的思维，以及有效的、适应环境的综合能力。
- 智能是个体认识客观事物和运用知识解决问题的能力。
- 人类个体的智能是一种综合能力。

1.1 人工神经网络的提出

- 智能可以包含8个方面
- 感知与认识客观事物、客观世界和自我的能力
  - 感知是智能的基础——最基本的能力
- 通过学习取得经验与积累知识的能力
  - 这是人类在世界中能够不断发展的最基本能力。
- 理解知识，运用知识和经验分析、解决问题的能力
  - 这一能力可以算作是智能的高级形式。是人类对世界进行适当的改造，推动社会不断发展的基本能力。

1.1 人工神经网络的提出

- 联想、推理、判断、决策语言的能力
  - 这是智能的高级形式的又一方面。
  - 预测和认识
  - “主动”和“被动”之分。联想、推理、判断、决策的能力是“主动”的基础。
- 运用进行抽象、概括的能力
- 上述这5种能力，被认为是人类智能最为基本的能力

1.1 人工神经网络的提出

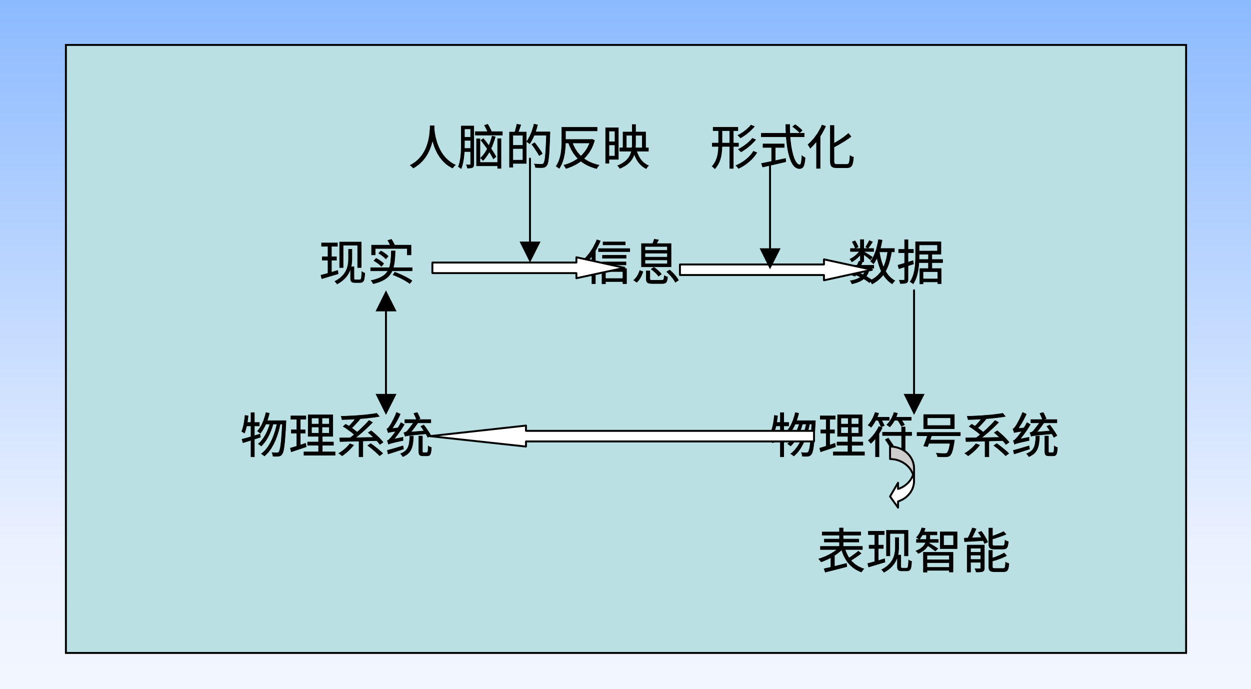
- 作为5种能力综合表现形式的3种能力
  - 发现、发明、创造、创新的能力
  - 实时、迅速、合理地应付复杂环境的能力
  - 预测、洞察事物发展、变化的能力

1.1 人工神经网络的提出

- 二、人工智能
- 人工智能：研究如何使类似计算机这样的设备去模拟人类的这些能力。
- 研究人工智能的目的
  - 增加人类探索世界，推动社会前进的能力
  - 进一步认识自己
- 三大学术流派
  - 符号主义（或叫做符号/逻辑主义）学派
  - 联接主义（或者叫做PDP）学派
  - 进化主义（或者叫做行动/响应）学派

1.1 人工神经网络的提出

- **1.1.2 物理符号系统**
-



1.1 人工神经网络的提出

- Newell和Simon假说 ：一个物理系统表现智能行为的充要条件是它有一个物理符号系统
- 概念：物理符号系统需要有一组称为符号的实体组成，它们都是物理模型，可以在另一类称为符号结构的实体中作为成分出现，以构成更高级别的系统

1.1 人工神经网络的提出

- 困难：
  - 抽象——舍弃一些特性，同时保留一些特性
  - 形式化处理——用物理符号及相应规则表达物理系统的存在和运行。
- 局限：
  - 对全局性判断、模糊信息处理、多粒度的视觉信息处理等是非常困难的。

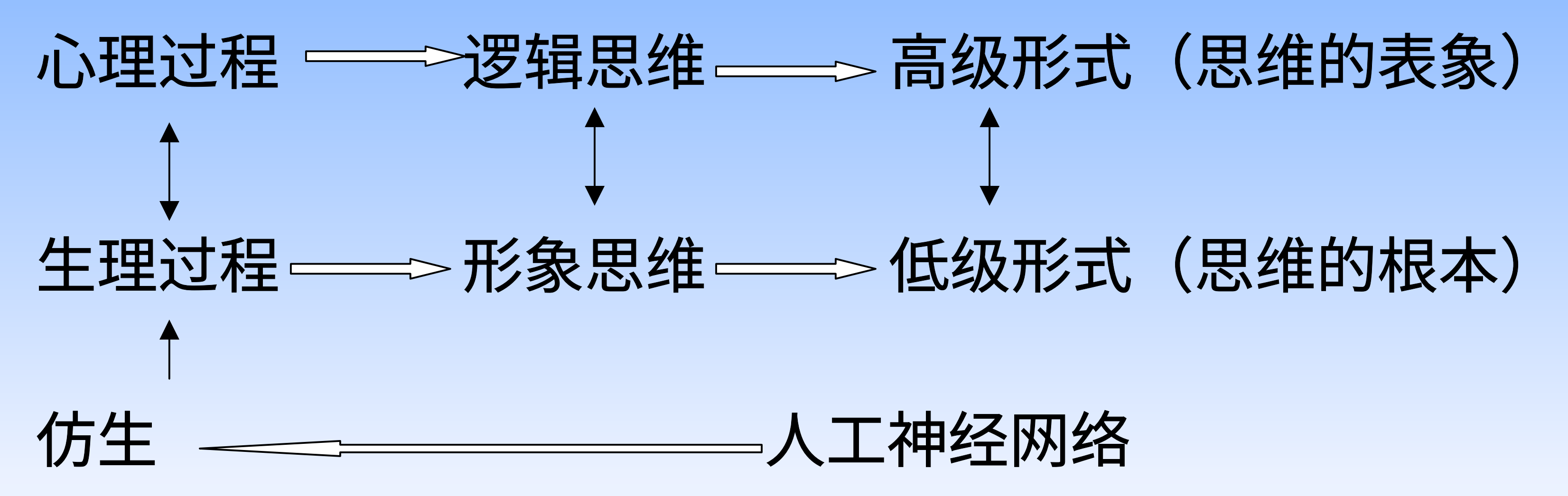
1.1 人工神经网络的提出

- 1.1.3　联接主义观点
- 核心：智能的本质是联接机制。
- 神经网络是一个由大量简单的处理单元组成的高度复杂的大规模非线性自适应系统
- ANN力求从四个方面去模拟人脑的智能行为
  - 物理结构
  - 计算模拟
  - 存储与操作
  - 训练

1.1 人工神经网络的提出

- 1.1.4　两种模型的比较

物理符号系统



联结主义观点

1.1 人工神经网络的提出

- 物理符号系统和人工神经网络系统的差别

项目	物理符号系统	人工神经网络
处理方式	逻辑运算	模拟运算
执行方式	串行	并行
动作	离散	连续
存储	局部集中	全局分布

1.1 人工神经网络的提出

- 两种人工智能技术的比较

项目	传统的AI技术	ANN技术
基本实现方式	串行处理；由程序实现控制	并行处理；对样本数据进行多目标学习；通过人工神经元之间的相互作用实现控制
基本开发方法	设计规则、框架、程序；用样本数据进行调试（由人根据已知的环境去构造一个模型）	定义人工神经网络的结构原型，通过样本数据，依据基本的学习算法完成学习——自动从样本数据中抽取内涵（自动适应应用环境）
适应领域	精确计算：符号处理，数值计算	非精确计算：模拟处理，感觉，大规模数据并行处理
模拟对象	左脑（逻辑思维）	右脑（形象思维）

1.2 人工神经网络的特点

- 信息的分布表示
- 运算的全局并行和局部操作
- 处理的非线性

1.2.1 人工神经网络的概念

1、定义

1）Hecht—Nielsen（1988年）

人工神经网络是一个并行、分布处理结构，它由处理单元及其称为联接的无向讯号通道互连而成。这些处理单元（**PE—Processing Element**）具有局部内存，并可以完成局部操作。每个处理单元有一个单一的输出联接，这个输出可以根据需要被分枝成希望个数的许多并行联接，且这些并行联接都输出相同的信号，即相应处理单元的信号，信号的大小不因分支的多少而变化。

1.2.1 人工神经网络的概念

（1）**Hecht—Nielsen（1988年）**（续）

- 处理单元的输出信号可以是任何需要的数学模型，每个处理单元中进行的操作必须是完全局部的。也就是说，它必须仅仅依赖于经过输入联接到达处理单元的所有输入信号的当前值和存储在处理单元局部内存中的值。

1.2.1 人工神经网络的概念

- 强调：
  - ① 并行、分布处理结构；
  - ② 一个处理单元的输出可以被任意分枝，且大小不变；
  - ③ 输出信号可以是任意的数学模型；
  - ④ 处理单元完全的局部操作

1.2.1 人工神经网络的概念

（2） **Rumellhart，McClelland，Hinton的PDP**

1） 一组处理单元（**PE或AN**）；

2） 处理单元的激活状态（a_i）；

3） 每个处理单元的输出函数（f_i）；

4） 处理单元之间的联接模式；

5） 传递规则（$\sum w_{ij}o_i$）；

6） 把处理单元的输入及当前状态结合起来产生激活值的激活规则（F_i）；

7） 通过经验修改联接强度的学习规则；

8） 系统运行的环境（样本集合）。

1.2.1 人工神经网络的概念

（3） Simpson（1987年）

- 人工神经网络是一个非线性的有向图，图中含有可以通过改变权大小来存放模式的加权边，并且可以从不完整的或未知的输入找到模式。

1.2.1 人工神经网络的概念

2、关键点

（1） 信息的分布表示

（2） 运算的全局并行与局部操作

（3） 处理的非线性特征

3、对大脑基本特征的模拟

1） 形式上：神经元及其联接；BN对AN

2） 表现特征：信息的存储与处理

1.2.1 人工神经网络的概念

4、别名

- 人工神经系统（ANS）
- 神经网络（NN）
- 自适应系统（Adaptive Systems）、自适应网（Adaptive Networks）
- 联接模型（Connectionism）
- 神经计算机（Neurocomputer）

1.2.2 学习（Learning）能力

- 人工神经网络可以根据所在的环境去改变它的行为
- 自相联的网络
- 异相联的网络：它在接受样本集合A时，可以抽取集合A中输入数据与输出数据之间的映射关系。——“抽象”功能。
- 不同的人工神经网络模型，有不同的学习/训练算法

1.2.3 基本特征的自动提取

- 由于其运算的不精确性，表现成"去噪音、容残缺"的能力，利用这种不精确性，比较自然地实现模式的自动分类。
- 普化（Generalization）能力与抽象能力

1.2.4 信息的分布存放

- 信息的分布存提供容错功能
  - 由于信息被分布存放在几乎整个网络中，所以，当其中的某一个点或者某几个点被破坏时，信息仍然可以被存取。
- 系统在受到局部损伤时还可以正常工作。
- 并不是说可以任意地对完成学习的网络进行修改。也正是由于信息的分布存放，对一类网来说，当它完成学习后，如果再让它学习新的东西，这时就会破坏原来已学会的东西。

1.2.5适应性(Applicability)问题

- 擅长两个方面：
  - 对大量的数据进行分类，并且只有较少的几种情况；
  - 必须学习一个复杂的非线性映射。
- 目前应用：
  - 人们主要将其用于语音、视觉、知识处理、辅助决策等方面。
  - 在数据压缩、模式匹配、系统建模、模糊控制、求组合优化问题的最佳解的近似解（不是最佳近似解）等方面也有较好的应用。

1.3 历史回顾

1.3.1 萌芽期（20世纪40年代）

- 人工神经网络的研究最早可以追溯到人类开始研究自己的智能的时期，到1949年止。
- 1943年，心理学家McCulloch和数学家Pitts建立起了著名的阈值加权和模型，简称为M-P模型。发表于数学生物物理学会刊《Bulletin of Methematical Biophysics》
- 1949年，心理学家D. O. Hebb提出神经元之间突触联系是可变的假说——Hebb学习律。

1.3.2 第一高潮期（1950~1968）

- 以 Marvin Minsky，Frank Rosenblatt，Bernard Widrow等为代表人物，代表作是单级感知器（Perceptron）。
- 可用电子线路模拟。
- 人们乐观地认为几乎已经找到了智能的关键。许多部门都开始大批地投入此项研究，希望尽快占领制高点。

1.3.3 反思期（1969~1982）

- M. L. Minsky和S. Papert，《Perceptron》，MIT Press，1969年
- 异或”运算不可表示
- 二十世纪70年代和80年代早期的研究结果
- 认识规律：认识——实践——再认识

1.3.4 第二高潮期（1983~1990）

- **1982年，J. Hopfield提出循环网络**
  - 用**Lyapunov函数**作为网络性能判定的能量函数，建立ANN稳定性的判别依据
  - 阐明了ANN与动力学的关系
  - 用非线性动力学的方法来研究ANN的特性
  - 指出信息被存放在网络中神经元的联接上

$$V=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}s(x_i)s(x_j)-\sum_{i=1}^{n}\int_0^{x_i}s_i'(\theta_i)\beta_i(\theta_i)d\theta_i-\sum_{i=1}^{n}\sum_{j-1}^{n}w_{ij}$$

1.3.4 第二高潮期（1983~1990）

- 2）1984年， J. Hopfield设计研制了后来被人们称为Hopfield网的电路。较好地解决了著名的TSP问题，找到了最佳解的近似解，引起了较大的轰动。
- 3）1985年，UCSD的Hinton、Sejnowsky、Rumelhart等人所在的并行分布处理（PDP）小组的研究者在Hopfield网络中引入了随机机制，提出所谓的Boltzmann机。

1.3.4 第二高潮期（1983~1990）

- 4）1986年，并行分布处理小组的Rumelhart等研究者重新独立地提出多层网络的学习算法——BP算法，较好地解决了多层网络的学习问题。（Paker1982和Werbos1974年）
- 国内首届神经网络大会是1990年12月在北京举行的。

1.3.5 再认识与应用研究期（1991~）

- 问题：
- 1）应用面还不够宽
- 2）结果不够精确
- 3）存在可信度的问题

1.3.5 再认识与应用研究期（1991~）

- 研究：
- 1）开发现有模型的应用，并在应用中根据实际运行情况对模型、算法加以改造，以提高网络的训练速度和运行的准确度。
- 2）充分发挥两种技术各自的优势是一个有效方法
- 3）希望在理论上寻找新的突破，建立新的专用/通用模型和算法。
- 4）进一步对生物神经系统进行研究，不断地丰富对人脑的认识。

第2章 人工神经网络基础

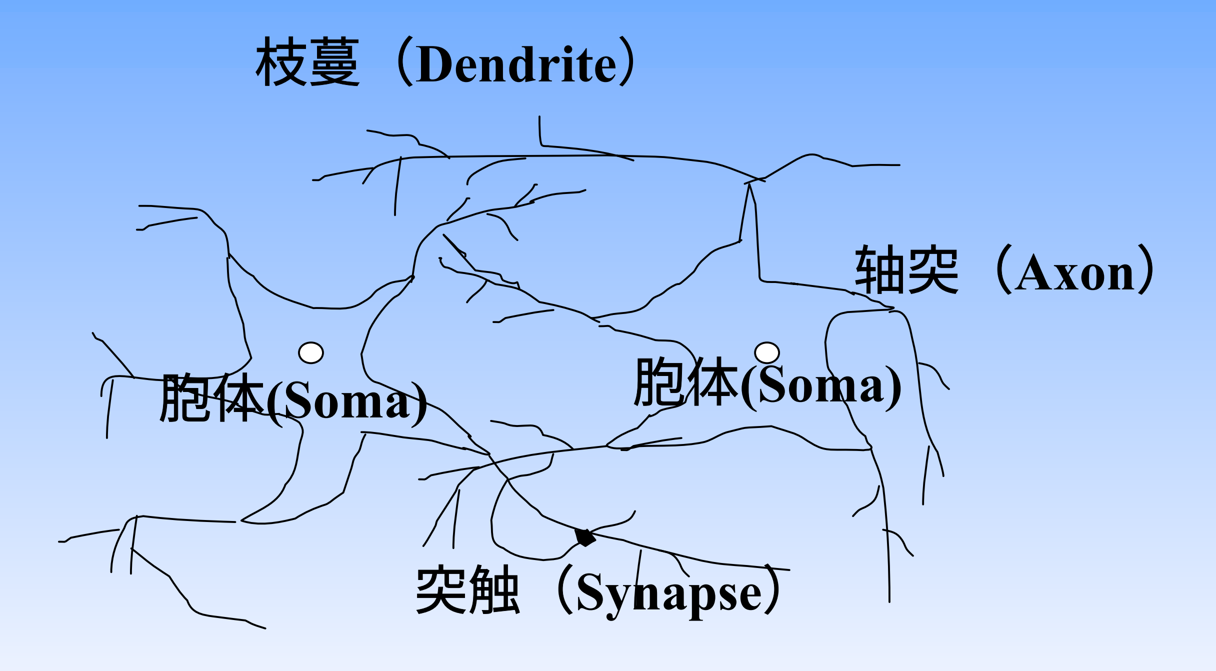
- 主要内容：
  - BN与AN；
  - 拓扑结构；
  - 存储；
  - 训练
- 重点：AN；拓扑结构；训练
- 难点：训练

第2章 人工神经网络基础

2.1 生物神经网
2.2 人工神经元
2.3 人工神经网络的拓扑特性
2.4 存储与映射
2.5 人工神经网络的训练

2.1 生物神经网

1、构成



2、工作过程

2.1 生物神经网

3、六个基本特征：

1）神经元及其联接；

2）神经元之间的联接强度决定信号传递的强弱；

3）神经元之间的联接强度是可以随训练改变的；

4）信号可以是起刺激作用的，也可以是起抑制作用的；

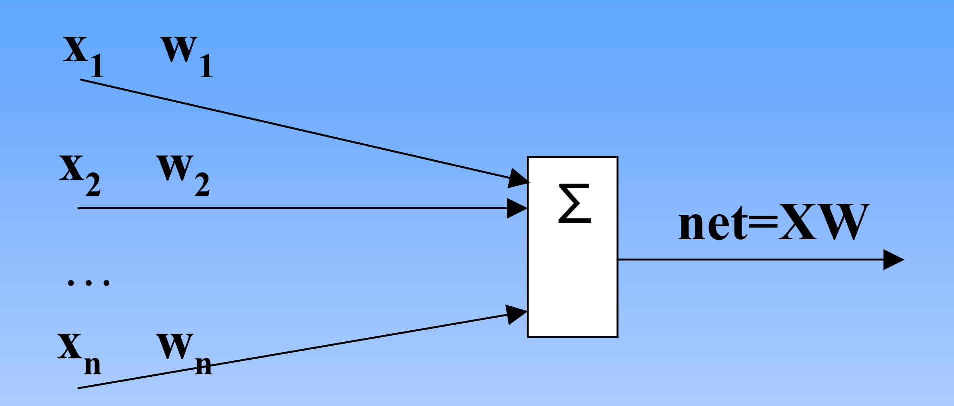
5）一个神经元接受的信号的累积效果决定该神经元的状态；

6) 每个神经元可以有一个"阈值"。

2.2 人工神经元

- 神经元是构成神经网络的最基本单元（构件）。
- 人工神经元模型应该具有生物神经元的六个基本特性。

2.2.1 人工神经元的基本构成



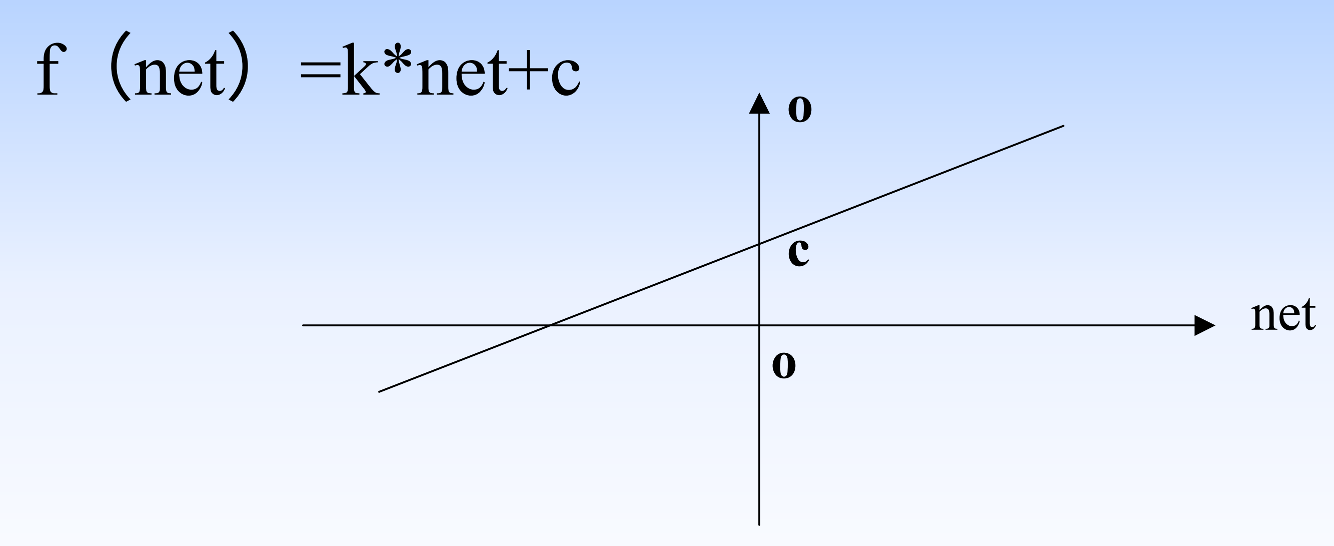
- 人工神经元模拟生物神经元的一阶特性。
  - 输入：$X=(x_1, x_2, \ldots, x_n)$
  - 联接权：$W=(w_1, w_2, \ldots, w_n)^T$
  - 网络输入：　$net=\sum x_i w_i$
  - 向量形式：　$net=XW$

2.2.2 激活函数(Activation Function)

- 激活函数——执行对该神经元所获得的网络输入的变换，也可以称为激励函数、活化函数： **o=f（net）**

1、线性函数（Liner Function）

f（net）=k*net+c

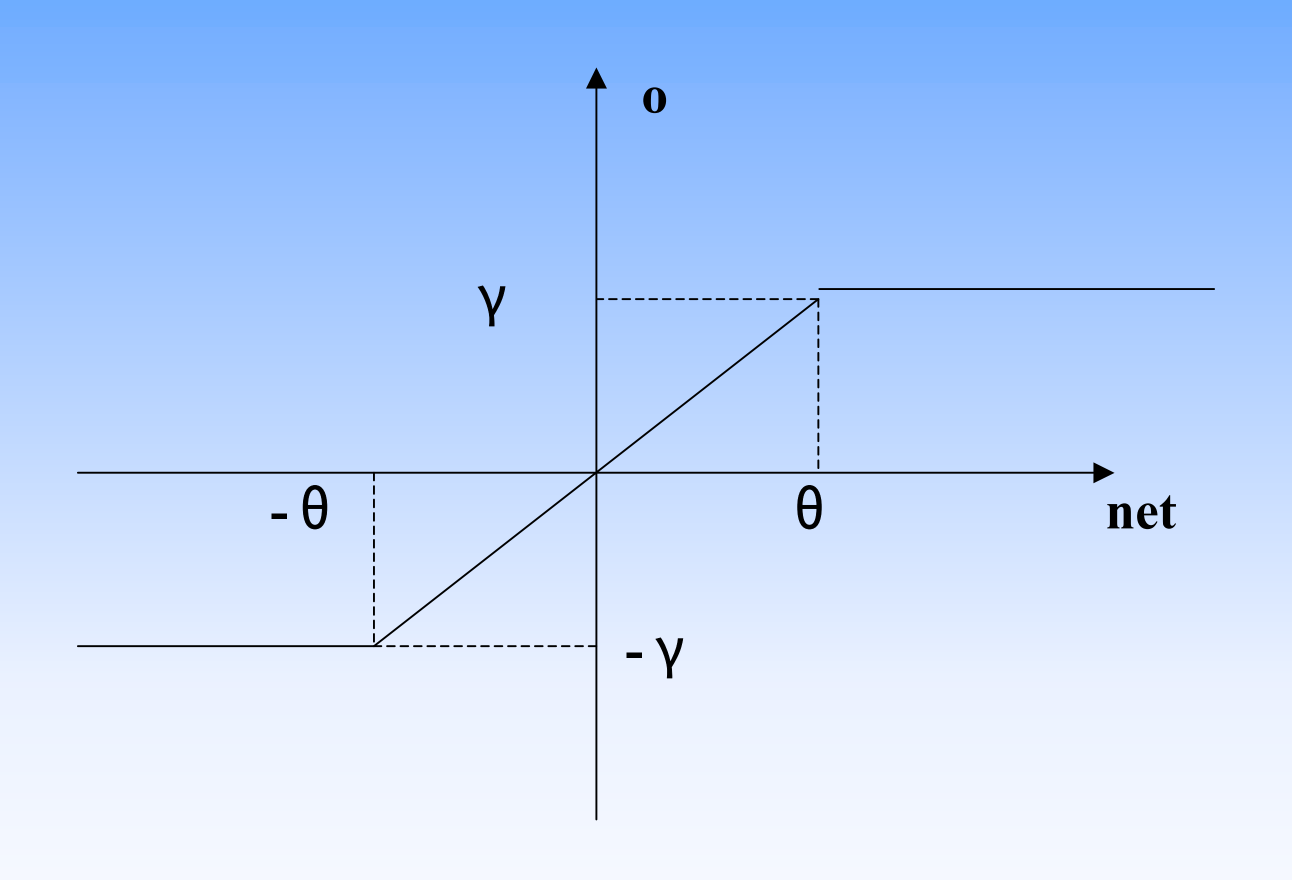


2、非线性斜面函数(Ramp Function)

$$f(net)=\begin{cases} \gamma & \text{if } net \geq \theta \\ k*net & \text{if } |net| < \theta \\ -\gamma & \text{if } net \leq -\theta \end{cases}$$

- $\gamma>0$为一常数，被称为饱和值，为该神经元的最大输出。

2、非线性斜面函数（Ramp Function）



3、阈值函数（Threshold Function）阶跃函数

$$f(net)=\begin{cases}\beta & \text{if } net>\theta \\ -\gamma & \text{if } net\le\theta\end{cases}$$

β、γ、θ均为非负实数，θ为阈值

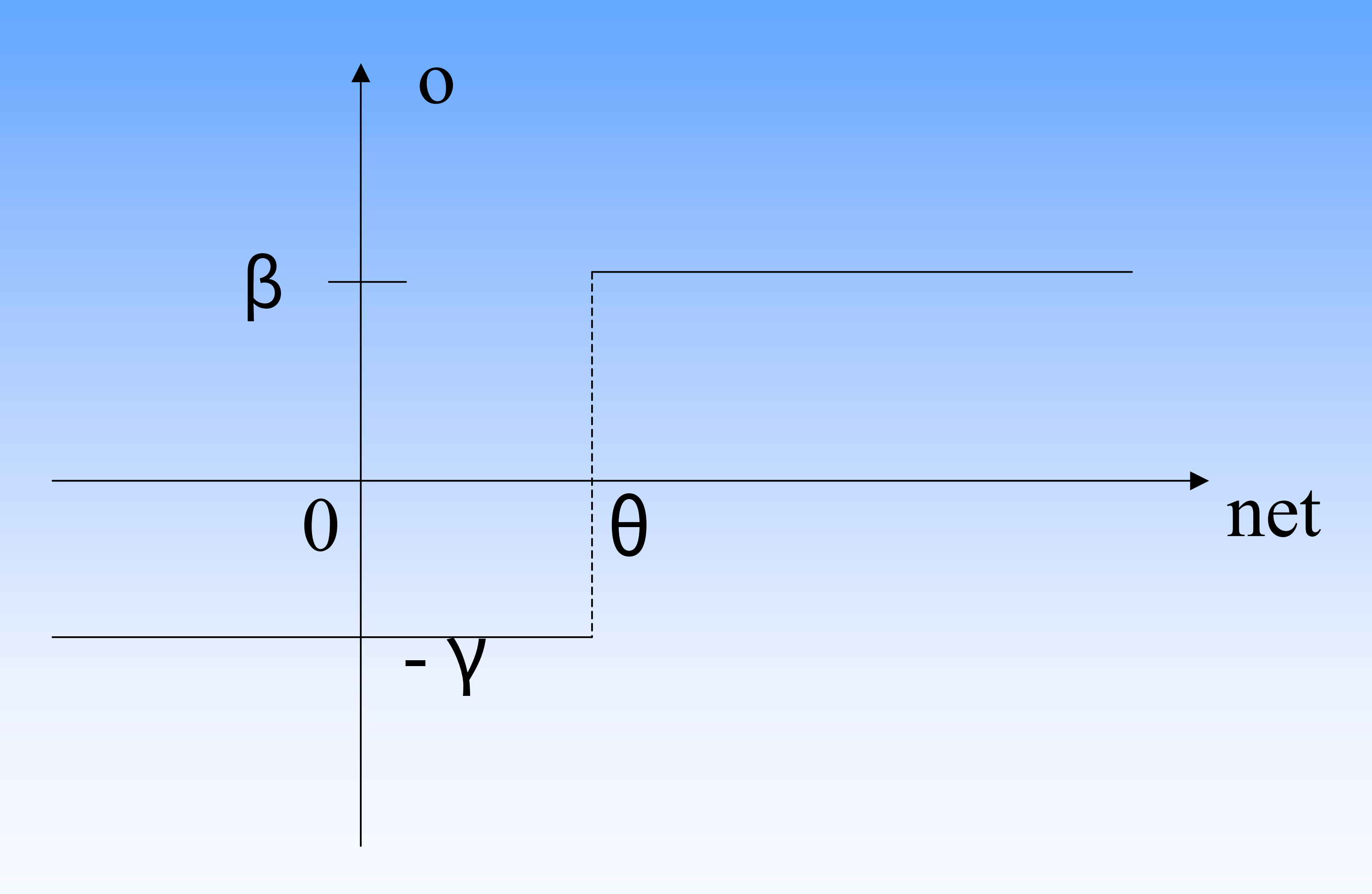
二值形式：

$$f(net)=\begin{cases}1 & \text{if } net>\theta \\ 0 & \text{if } net\le\theta\end{cases}$$

双极形式：

$$f(net)=\begin{cases}1 & \text{if } net>\theta \\ -1 & \text{if } net\le\theta\end{cases}$$

3、阈值函数（Threshold Function）阶跃函数



4、S形函数

压缩函数（Squashing Function）和逻辑斯特函数（Logistic Function）。

f（net）=a+b/(1+exp(-d*net))

a，b，d为常数。它的饱和值为a和a+b。

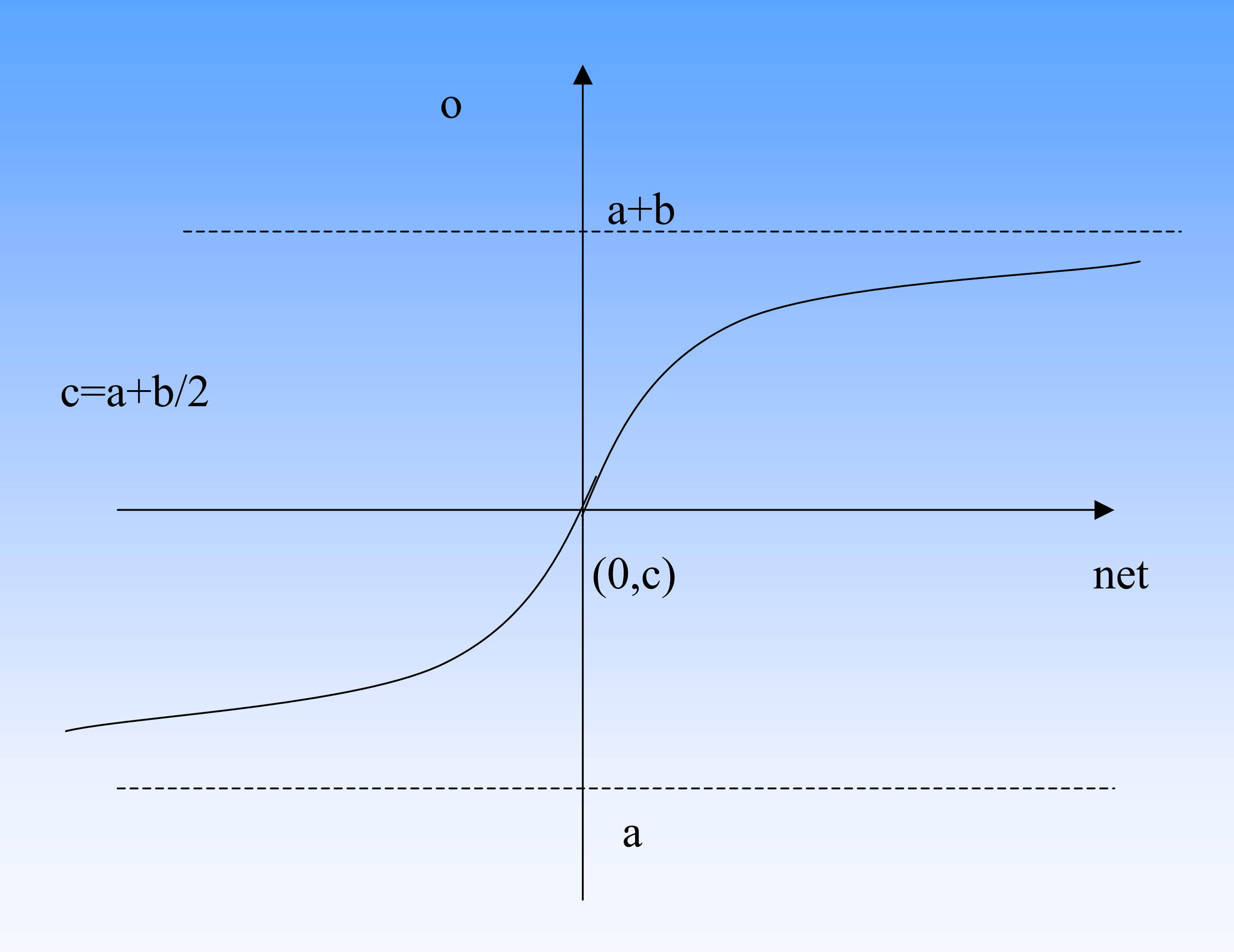
最简单形式为：

f（net）= 1/(1+exp(-d*net))

函数的饱和值为0和1。

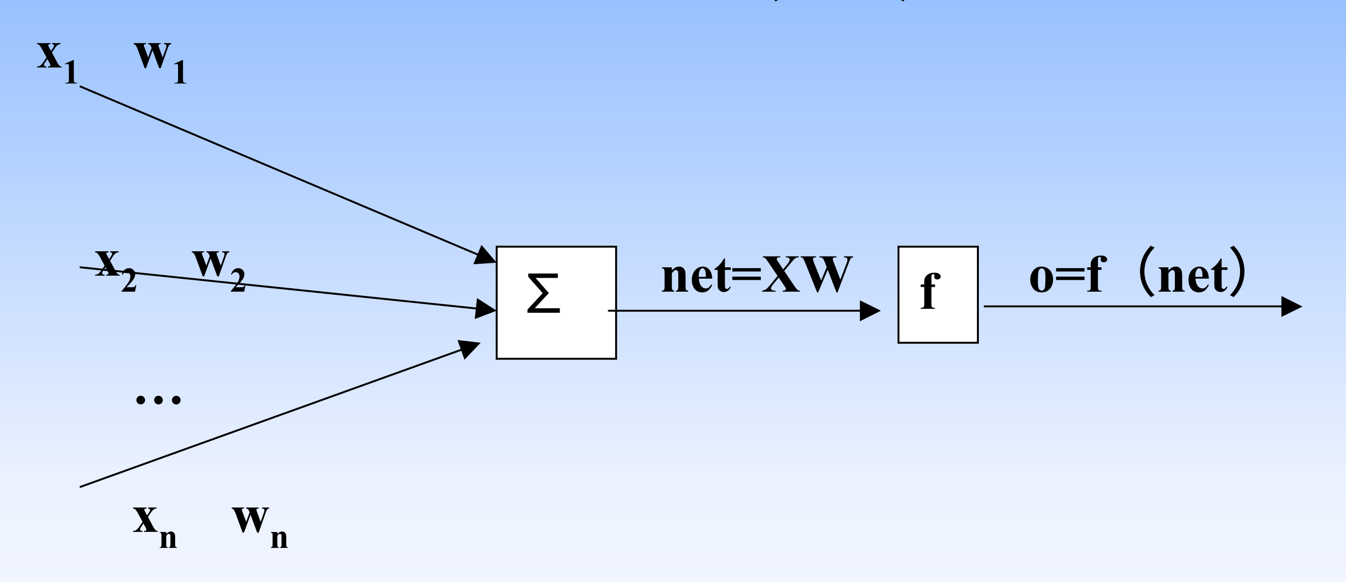
- S形函数有较好的增益控制

4、S形函数



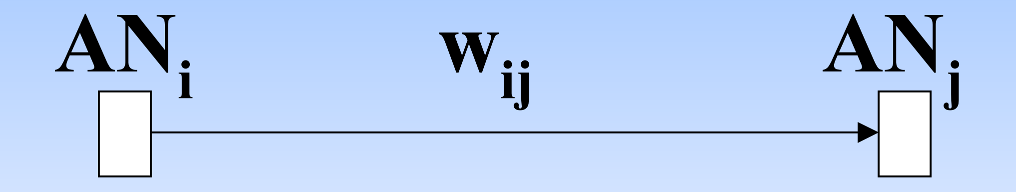
2.2.3 M-P模型

McCulloch—Pitts（M—P）模型，也称为处理单元（PE）



2.3 人工神经网络的拓扑特性

连接的拓扑表示



2.3.1 联接模式

- 用正号（“+”，可省略）表示传送来的信号起刺激作用，它用于增加神经元的活跃度；
- 用负号（“-”）表示传送来的信号起抑制作用，它用于降低神经元的活跃度。
- 层次（又称为“级”）的划分，导致了神经元之间的三种不同的互连模式：

2.3.1 联接模式

- 1、 层（级）内联接
  - 层内联接又叫做区域内（Intra-field）联接或侧联接（Lateral）。
  - 用来加强和完成层内神经元之间的竞争
- 2、 循环联接
  - 反馈信号。

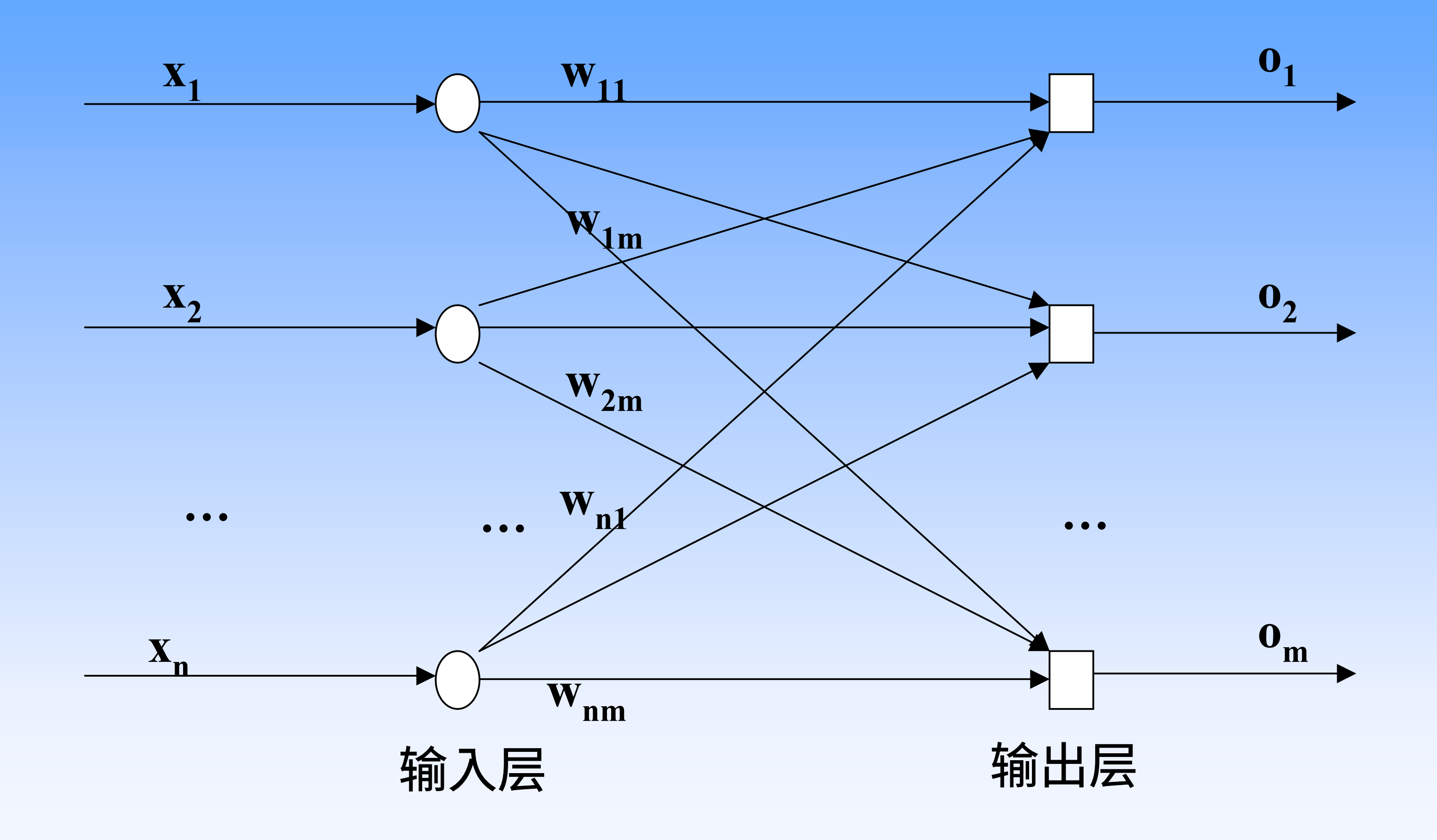
2.3.1 联接模式

- 3、层（级）间联接
  - 层间（Inter-field）联接指不同层中的神经元之间的联接。这种联接用来实现层间的信号传递
  - 前馈信号
  - 反馈信号

2.3.2 网络的分层结构

- 单级网
  - 简单单级网

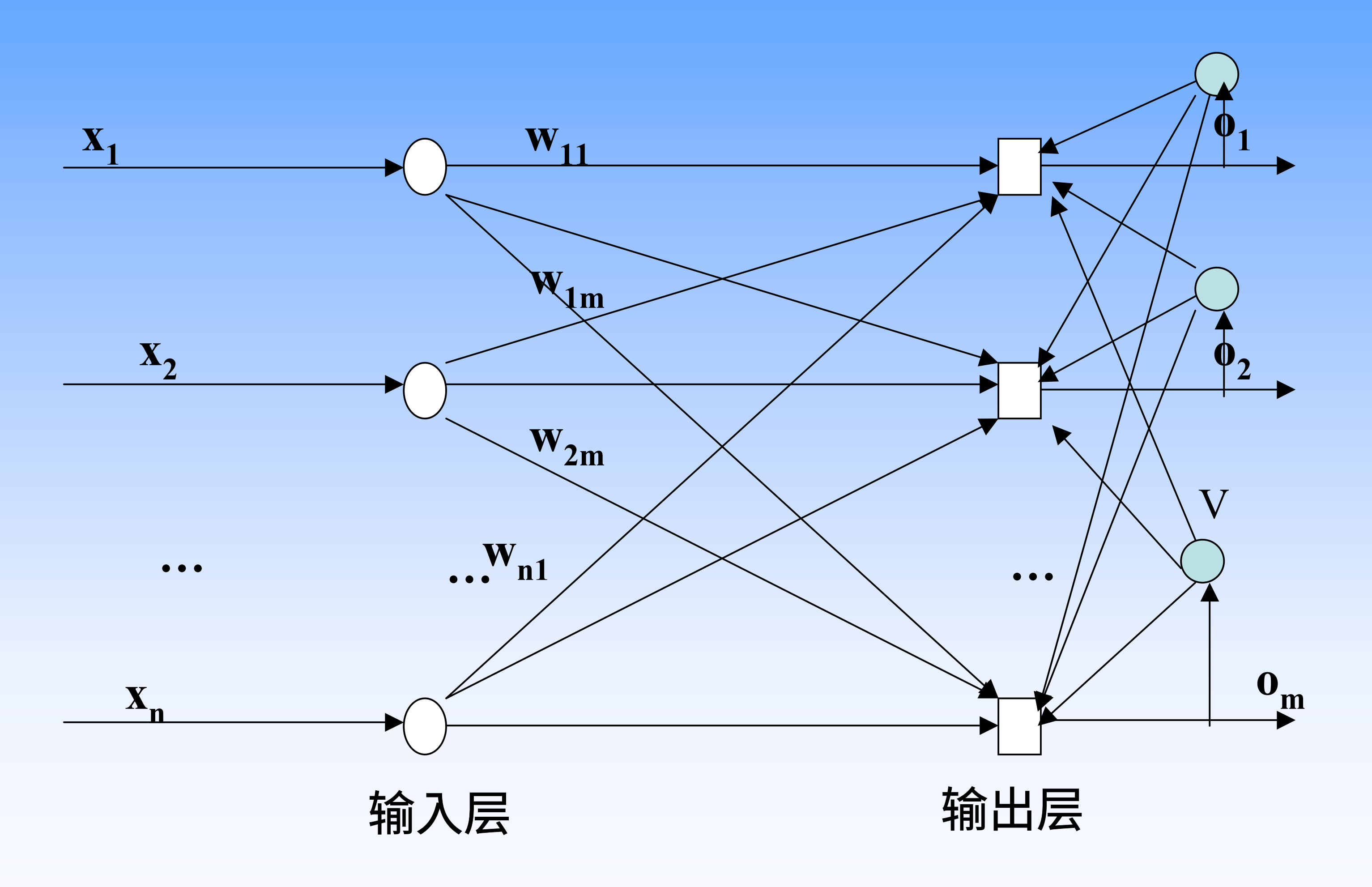
简单单级网



简单单级网

- $W=(w_{ij})$
- 输出层的第j个神经元的网络输入记为net_j：
- $net_j=x_1w_{1j}+x_2w_{2j}+\ldots+x_nw_{nj}$
- 其中, $1\leq j\leq m$。取
- $NET=(net_1，net_2，\ldots，net_m)$
- $NET=XW$
- $O=F(NET)$

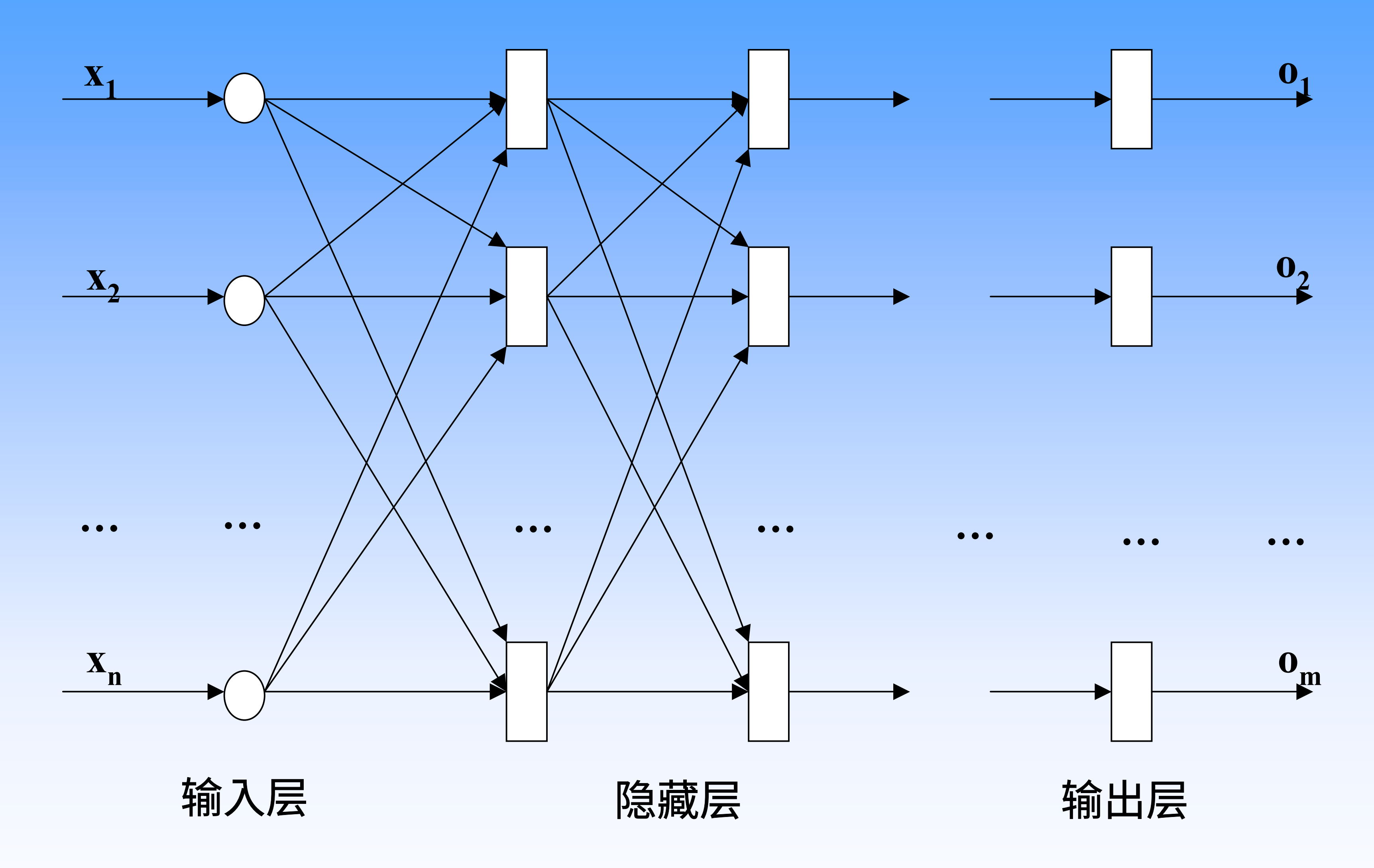
单级横向反馈网



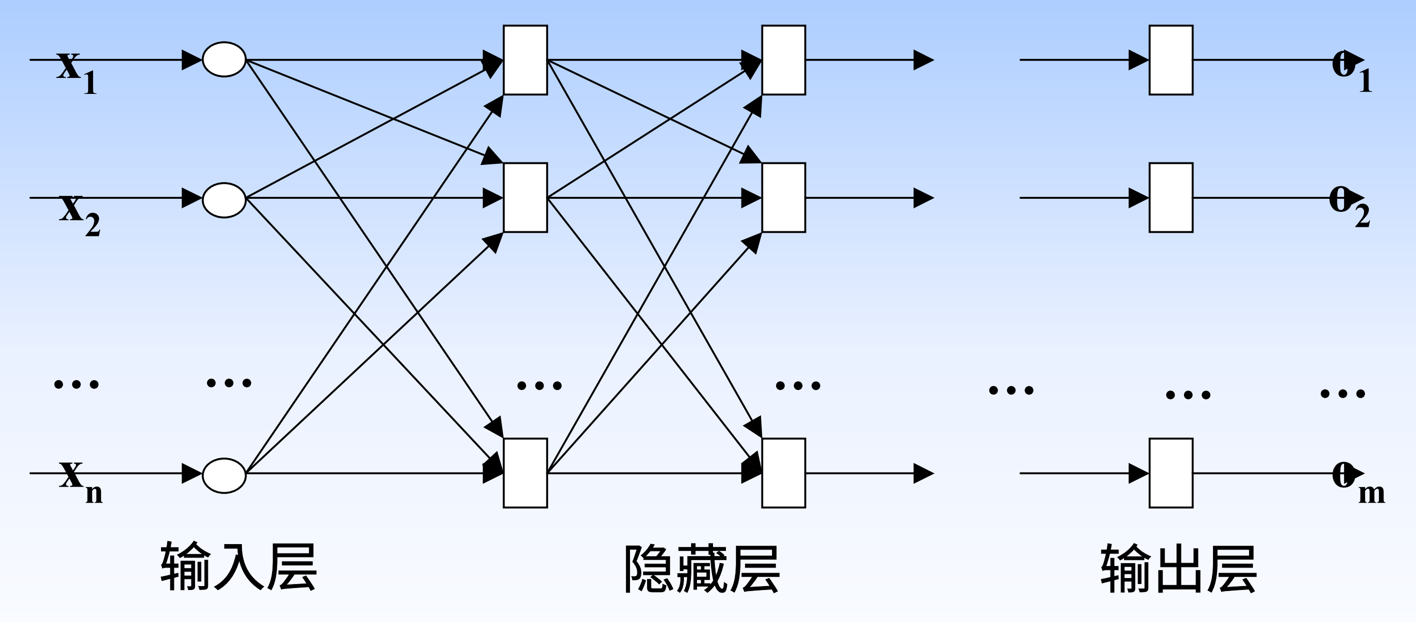
单级横向反馈网

- V=（v_{ij}）
- NET=XW+OV
- O=F（NET）
- 时间参数——神经元的状态在主时钟的控制下同步变化
- 考虑X总加在网上的情况
  - NET（t+1）=X（t）W+O（t）V
  - O(t+1)=F(NET(t+1))
    - O（0）=0
- 考虑仅在t=0时加X的情况。
- 稳定性判定

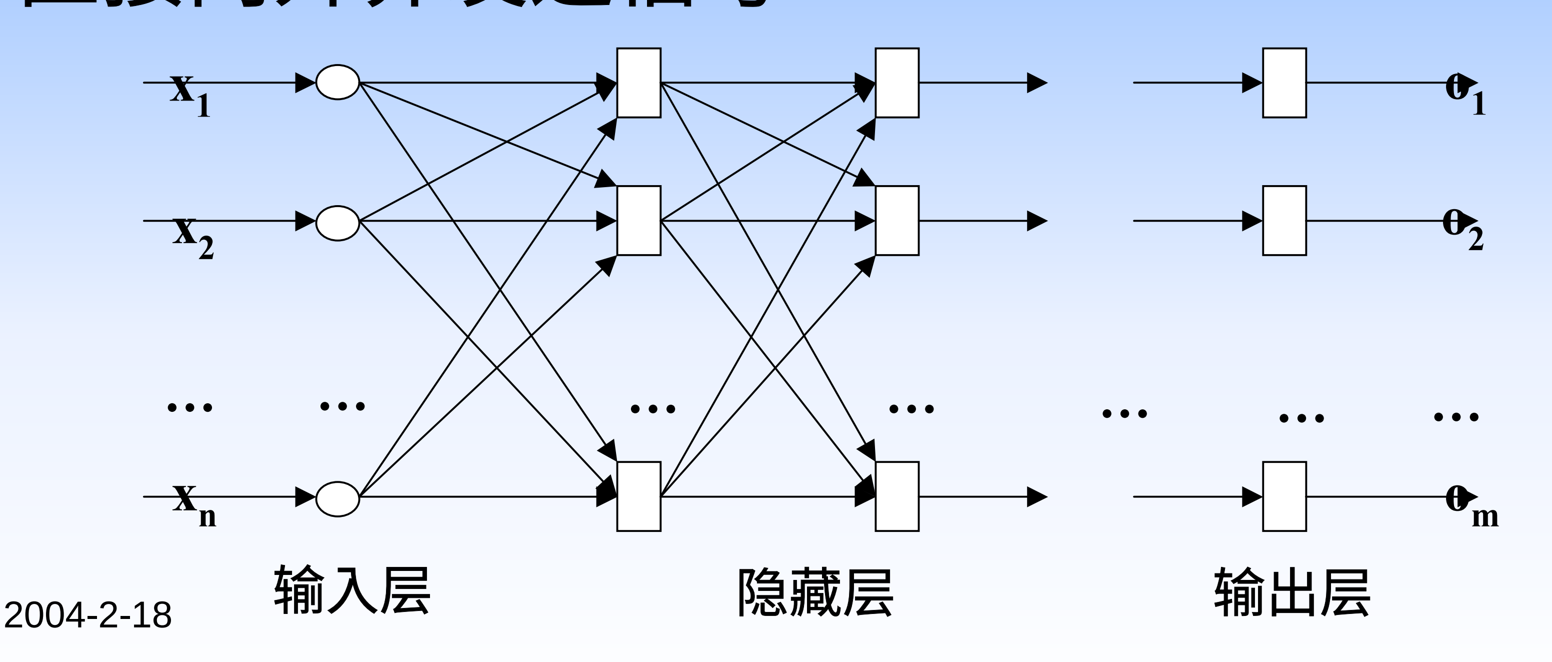
多级网



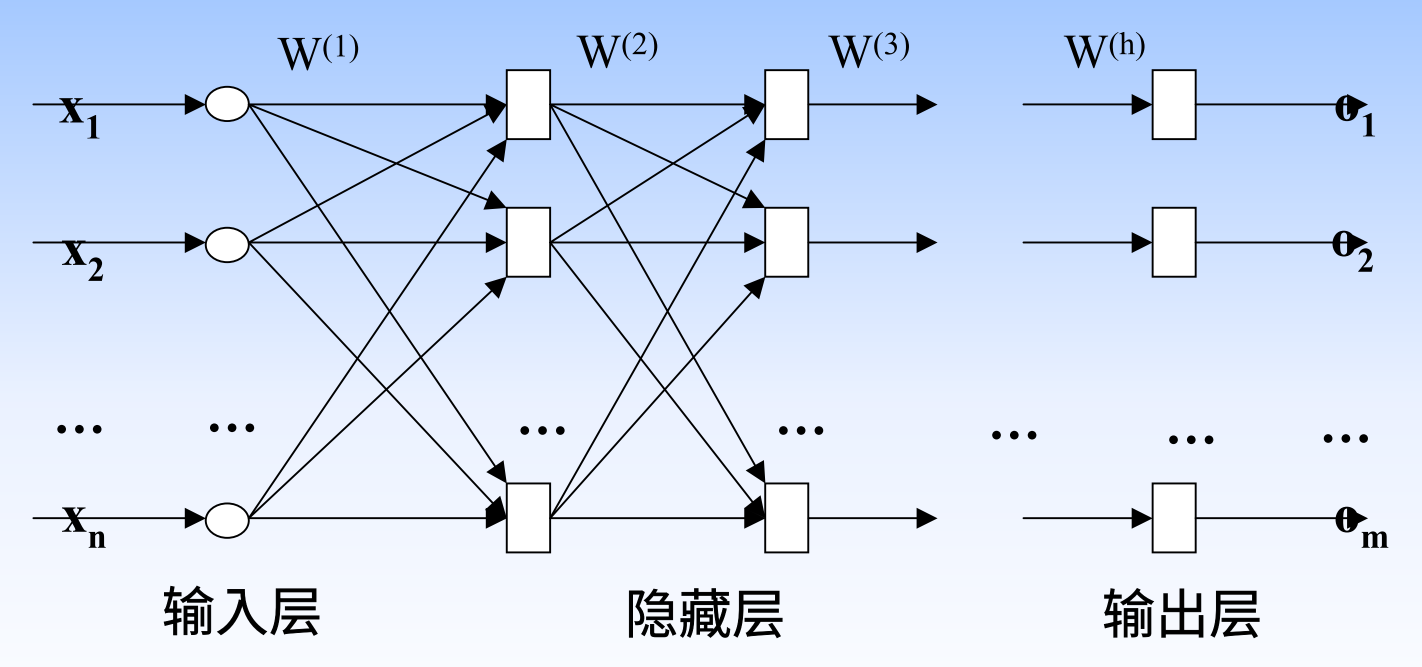
- 层次划分
  - 信号只被允许从较低层流向较高层。
  - 层号确定层的高低：层号较小者，层次较低，层号较大者，层次较高。
  - 输入层：被记作第0层。该层负责接收来自网络外部的信息



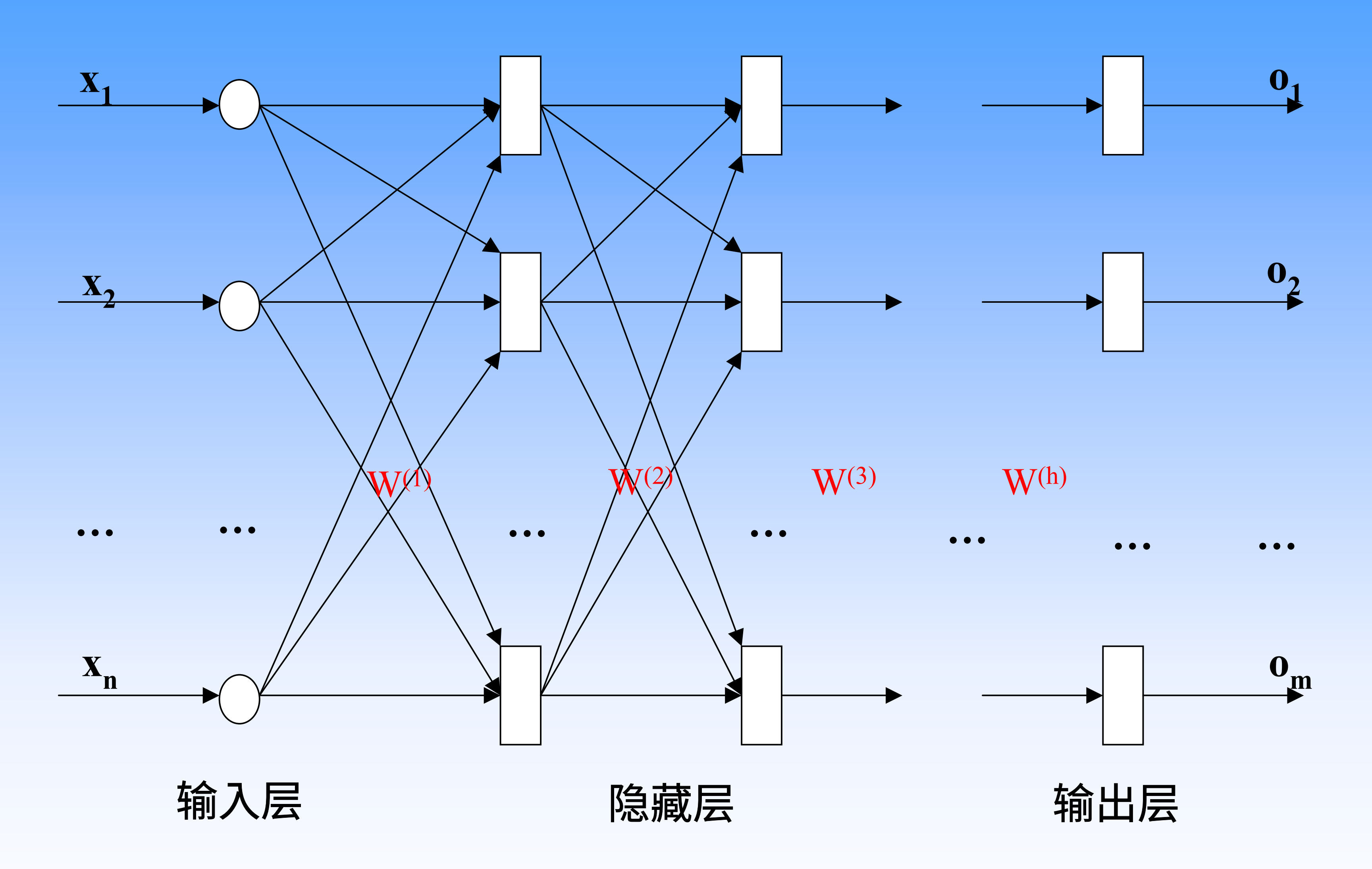
– 第j层：第j-1层的直接后继层（j>0），它直接接受第j-1层的输出。
– 输出层：它是网络的最后一层，具有该网络的最大层号，负责输出网络的计算结果。
– 隐藏层：除输入层和输出层以外的其它各层叫隐藏层。隐藏层不直接接受外界的信号，也不直接向外界发送信号



- 约定：
  - 输出层的层号为该网络的层数：n层网络，或n级网络。
  - 第j-1层到第j层的联接矩阵为第j层联接矩阵，输出层对应的矩阵叫输出层联接矩阵。今后，在需要的时候，一般我们用$W^{(j)}$表示第j层矩阵。



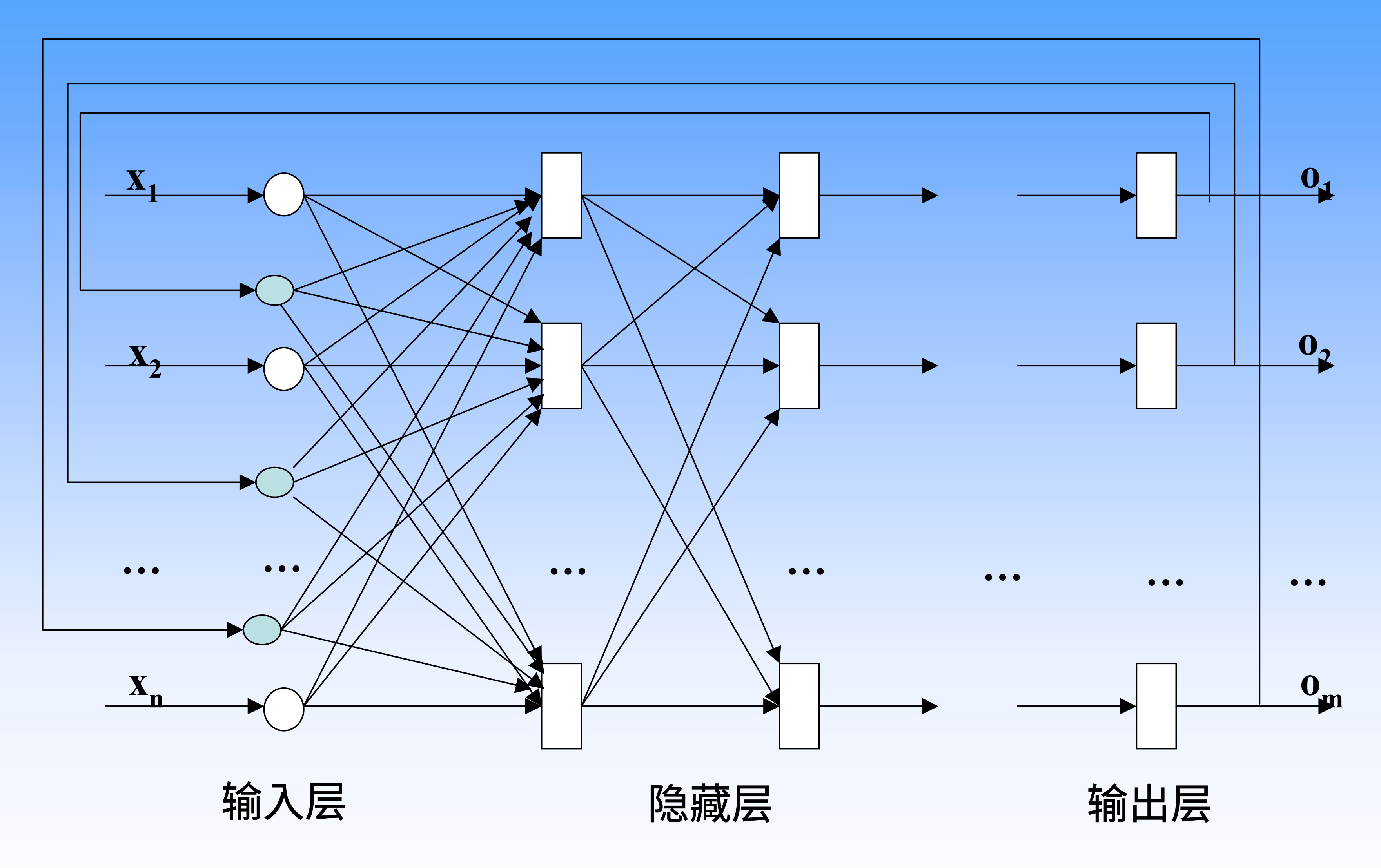
多级网——h层网络



多级网

- 非线性激活函数
  - $F(X)=kX+C$
  - $F_3(F_2(F_1(XW^{(1)})W^{(2)})W^{(3)})$

循环网



循环网

- 如果将输出信号反馈到输入端,就可构成一个多层的循环网络。
- 输入的原始信号被逐步地“加强”、被“修复”。
- 大脑的短期记忆特征——看到的东西不是一下子就从脑海里消失的。
- 稳定：反馈信号会引起网络输出的不断变化。我们希望这种变化逐渐减小，并且最后能消失。当变化最后消失时，网络达到了平衡状态。如果这种变化不能消失，则称该网络是不稳定的。

2.4 存储与映射

- 空间模式（Spatial Model）
- 时空模式（Spatialtemporal Model）
- 空间模式三种存储类型
- **1、 RAM方式（Random Access Memory）**
  - 随机访问方式是将地址映射到数据。
- **2、 CAM方式（Content Addressable Memory）**
  - 内容寻址方式是将数据映射到地址。
- **3、 AM方式（Associative Memory）**
  - 相联存储方式是将数据映射到数据。

2.4 存储与映射

- 后续的两种方式是人工神经网络的工作方式。
- 在学习/训练期间，人工神经网络以**CAM**方式工作；权矩阵又被称为网络的长期存储（**Long Term Memory**，简记为**LTM**）。
- 网络在正常工作阶段是以**AM**方式工作的；神经元的状态表示的模式为短期存储（**Short Term Memory**，简记为**STM**）。

2.4 存储与映射

- 自相联（Auto-associative）映射：训练网络的样本集为向量集合为

$\{A_1, A_2, \ldots, A_n\}$

- 在理想情况下，该网络在完成训练后，其权矩阵存放的将是上面所给的向量集合。

2.4 存储与映射

- 异相联（**Hetero-associative**）映射

{（A_1，B_1），（A_2，B_2），…，（A_n，B_n）}

- 该网络在完成训练后，其权矩阵存放的将是上面所给的向量集合所蕴含的对应关系。
- 当输入向量**A**不是样本的第一的分量时，样本中不存在这样的元素（A_k，B_k），使得

$$A_i \leq A_k \leq A \text{或者} A \leq A_k \leq A_j$$

- 且此时有

$$A_i \leq A \leq A_j$$

- 则向量**B**是B_i与B_j的插值。

2.5 人工神经网络的训练

- 人工神经网络最具有吸引力的特点是它的学习能力。
- 1962年，Rosenblatt给出了人工神经网络著名的学习定理：人工神经网络可以学会它可以表达的任何东西。
- 人工神经网络的表达能力大大地限制了它的学习能力。
- 人工神经网络的学习过程就是对它的训练过程

2.5.1无导师学习

- 无导师学习(Unsupervised Learning)与无导师训练(Unsupervised Training)相对应
- 抽取样本集合中蕴含的统计特性，并以神经元之间的联接权的形式存于网络中。

2.5.1无导师学习

- **Hebb学习律、竞争与协同（Competitive and Cooperative）学习、随机联接系统（Randomly Connected Learning）等。**
- **Hebb算法[D. O. Hebb在1961年]的核心：**
  - 当两个神经元同时处于激发状态时被加强，否则被减弱。
  - 数学表达式表示：
    - $W_{ij}(t+1)=W_{ij}(t)+\alpha o_i(t)\ o_j(t)$

2.5.2 有导师学习

- 有导师学习(Supervised Learning)与有导师训练(Supervised Training)相对应。
- 输入向量与其对应的输出向量构成一个“训练对”。

训练算法的主要步骤

1） 从样本集合中取一个样本（A_i，B_i）；
2） 计算出网络的实际输出O；
3） 求$D=B_i-O$；
4） 根据D调整权矩阵W；
5） 对每个样本重复上述过程，直到对整个样本集来说，误差不超过规定范围。

Delta规则

Widrow和Hoff的写法：

$$W_{ij}(t+1)=W_{ij}(t)+\alpha(y_j-a_j(t))o_i(t)$$

也可以写成：

$$W_{ij}(t+1)=W_{ij}(t)+\Delta W_{ij}(t)$$

$$\Delta W_{ij}(t)=\alpha\delta_j o_i(t)$$

$$\delta_j=y_j-a_j(t)$$

Delta规则

Grossberg的写法为：

$$\Delta W_{ij}(t)=\alpha a_i(t)(o_j(t)-W_{ij}(t))$$

更一般的**Delta**规则为：

$$\Delta W_{ij}(t)=g(a_i(t),\ y_j,\ o_j(t),\ W_{ij}(t))$$

第3章 感知器

- 主要内容：
  - 感知器与人工神经网络的早期发展；
  - 线性可分问题与线性不可分问题；
  - Hebb学习律；
  - Delta规则;
  - 感知器的训练算法。
- 重点：感知器的结构、表达能力、学习算法
- 难点：感知器的表达能力

第3章 感知器

3.1 感知器与人工神经网络的早期发展

3.2 感知器的学习算法

3.2.1 离散单输出感知器训练算法

3.2.2 离散多输出感知器训练算法

3.2.3 连续多输出感知器训练算法

实现！

3.3 线性不可分问题

3.3.1 异或(Exclusive –OR)问题

3.3.2 线性不可分问题的克服

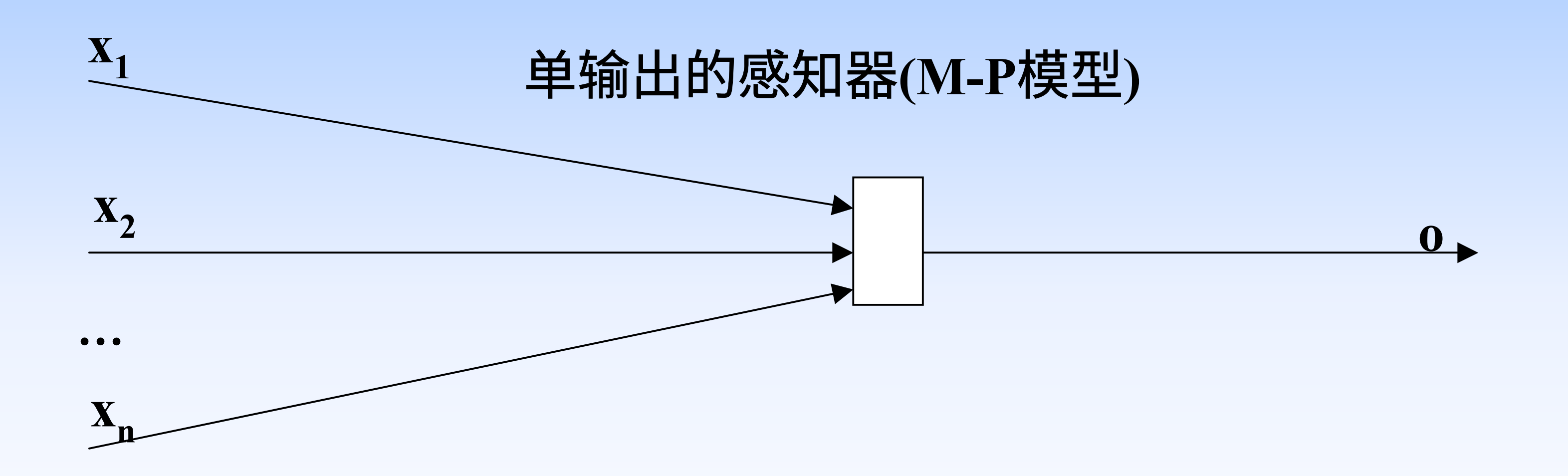
问题的发现与解决！

3.1 感知器与ANN的早期发展

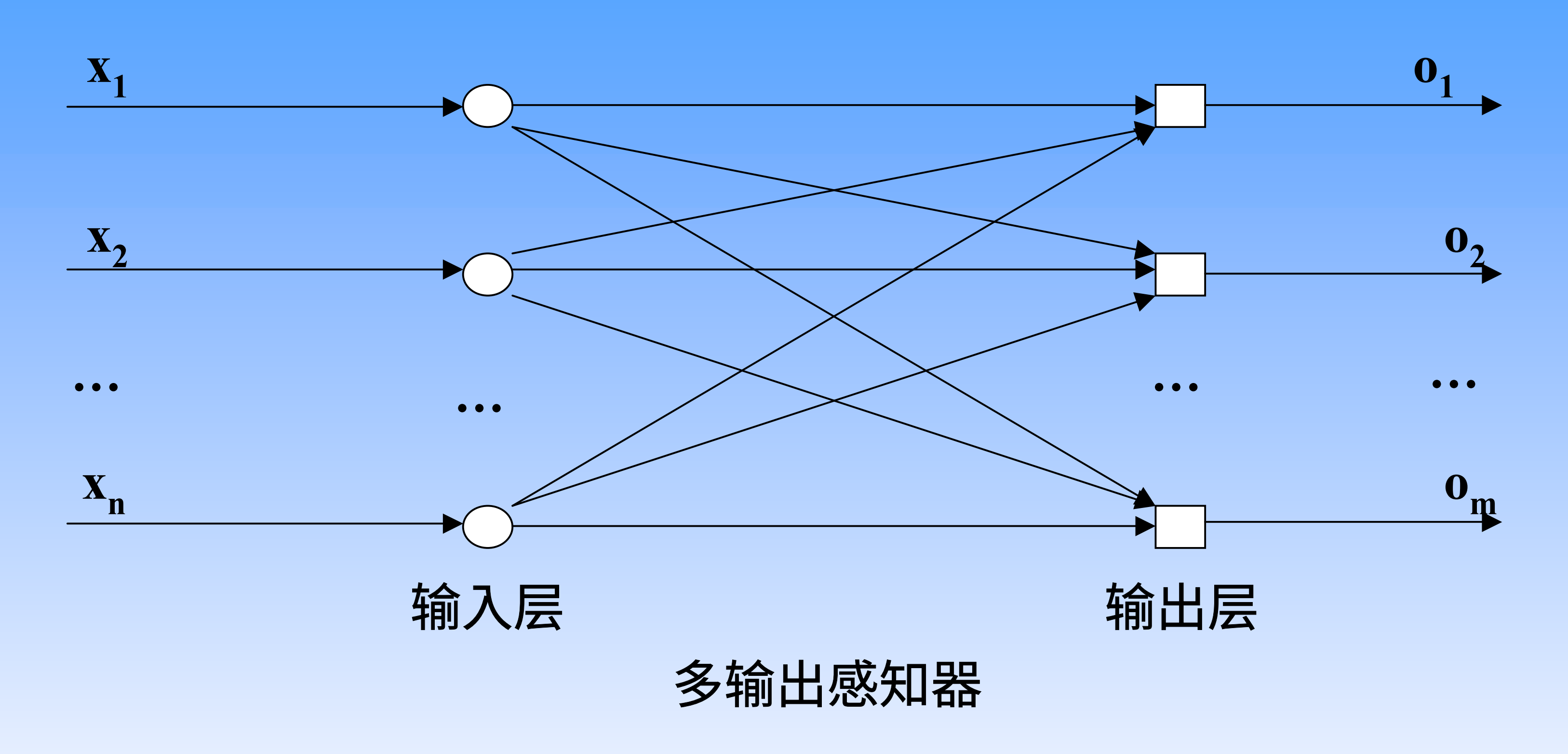
McCulloch 和**Pitts** **1943**年，发表第一个系统的**ANN**研究——阈值加权和(**M-P**)数学模型。

1947年，开发出感知器。

1949年，提出**Hebb**学习律。



3.1 感知器与ANN的早期发展



多输出感知器

- 1962年，Rosenblatt宣布：人工神经网络可以学会它能表示的任何东西

3.2 感知器的学习算法

- 感知器的学习是有导师学习
- 感知器的训练算法的基本原理来源于著名的**Hebb**学习律
- 基本思想：逐步地将样本集中的样本输入到网络中,根据输出结果和理想输出之间的差别来调整网络中的权矩阵

3.2.1离散单输出感知器训练算法

- 二值网络：自变量及其函数的值、向量分量的值只取0和1函数、向量。
- 权向量：$W=(w_1, w_2, \ldots, w_n)$
- 输入向量：$X=(x_1, x_2, \ldots, x_n)$
- 训练样本集：
  - {(X，Y)|Y为输入向量X对应的输出}

算法3-1离散单输出感知器训练算法

1. 初始化权向量W；

2. 重复下列过程，直到训练完成：

    2.1 对每个样本（X，Y），重复如下过程：

        2.1.1 输入X；

        2.1.2 计算o=F（XW）；

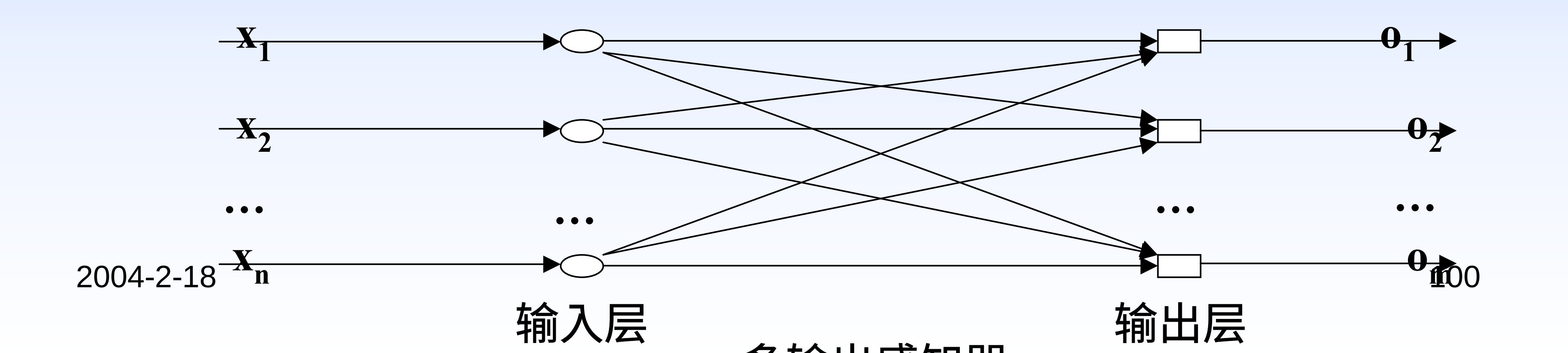
        2.1.3 如果输出不正确，则

            当o=0时，取 W=W+X，

            当o=1时，取 W=W-X

3.2.2离散多输出感知器训练算法

- 样本集：{(X, Y)|Y为输入向量X对应的输出}
- 输入向量：$X=(x_1, x_2, \ldots, x_n)$
- 理想输出向量：$Y=(y_1, y_2, \ldots, y_m)$
- 激活函数：F
- 权矩阵$W=(w_{ij})$
- 实际输出向量：$O=(o_1, o_2, \ldots, o_m)$



算法3-2离散多输出感知器训练算法

1.初始化权矩阵W；

2.重复下列过程，直到训练完成：

2.1 对每个样本（X，Y），重复如下过程：

2.1.1 输入X；

2.1.2 计算O=F（XW）；

2.1.3 for j=1 to m do 执行如下操作：

if $o_j \neq y_j$ then

if $o_i = 0$ then for i = 1 to n

$w_{ij}=w_{ij}+x_i$

else for i= 1 to n do

$w_{ij}=w_{ij}-x_i$

算法3-2离散多输出感知器训练算法

- 算法思想：将单输出感知器的处理逐个地用于多输出感知器输出层的每一个神经元的处理。
- 第1步，权矩阵的初始化：一系列小伪随机数。

算法3-2离散多输出感知器训练算法

- 第2步，循环控制。
- 方法1：循环次数控制法：对样本集执行规定次数的迭代
- 改进——分阶段迭代控制：设定一个基本的迭代次数N，每当训练完成N次迭代后，就给出一个中间结果

算法3-2离散多输出感知器训练算法

- 方法2：精度控制法：给定一个精度控制参数
  - 精度度量：实际输出向量与理想输出向量的对应分量的差的绝对值之和；
  - 实际输出向量与理想输出向量的欧氏距离的和
  - “死循环”：网络无法表示样本所代表的问题

算法3-2离散多输出感知器训练算法

- 方法3：综合控制法：将这两种方法结合起来使用
- 注意：精度参数的设置。根据实际问题选定；初始测试阶段，精度要求低，测试完成后，再给出实际的精度要求。

3.2.3 连续多输出感知器训练算法

- 用公式$w_{ij}=w_{ij}+\alpha(y_j-o_j)x_i$取代了算法3-2第2.1.3步中的多个判断
- y_j与o_j之间的差别对w_{ij}的影响由$\alpha(y_j-o_j)x_i$表现出来
- 好处：不仅使得算法的控制在结构上更容易理解，而且还使得它的适应面更宽

算法3-3 连续多输出感知器训练算法

1．用适当的小伪随机数初始化权矩阵W；

2. 初置精度控制参数ε，学习率α，精度控制变量d=ε+1；

3．While d ≥ ε do

  3.1 d=0；

  3.2 for 每个样本（X，Y）do

    3.2.1 输入X（$=(x_1, x_2, \ldots, x_n)$）；

    3.2.2 求O=F（XW）；

    3.2.3 修改权矩阵W：

      for i=1 to n，j=1 to m do

        $w_{ij}=w_{ij}+\alpha(y_j-o_j)x_i$；

    3.2.4 累积误差

      for j = 1 to m do

        $d=d+(y_j-o_j)^2$

算法3-3 连续多输出感知器训练算法

1、程序实现：ε、 α、d、i、j、n、m为简单变量来表示，W为n行m列的二维数组。样本集二维数组

2、系统的调试

3、Minsky在1969年证明，有许多基本问题是感知器无法解决

4、问题线性可分性可能与时间有关

5、很难从样本数据集直接看出问题是否线性可分

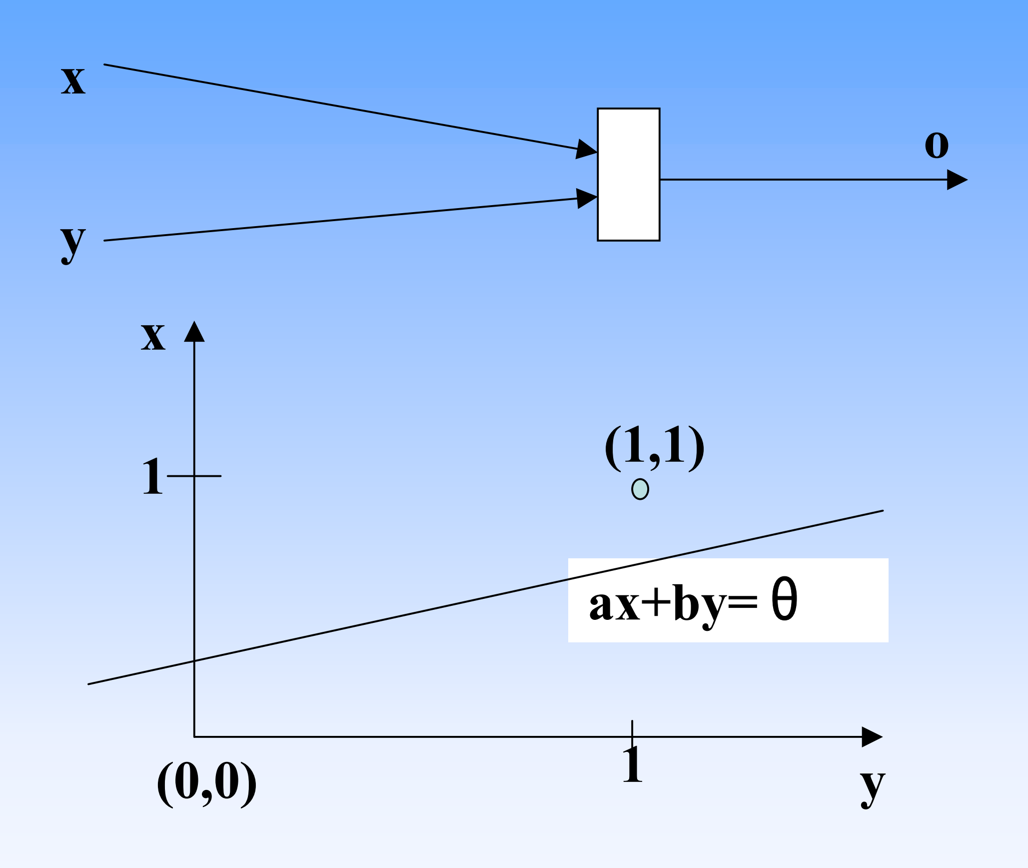
6、未能证明，一个感知器究竟需要经过多少步才能完成训练。

3.3 线性不可分问题

3.3.1 异或(Exclusive –OR)问题

g（x，y）		y	
		0	1
x	0	0	1
	1	1	0

用于求解XOR的单神经元感知器



单神经元感知器的图像

线性不可分函数

变量		函数及其值															
x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

线性不可分函数

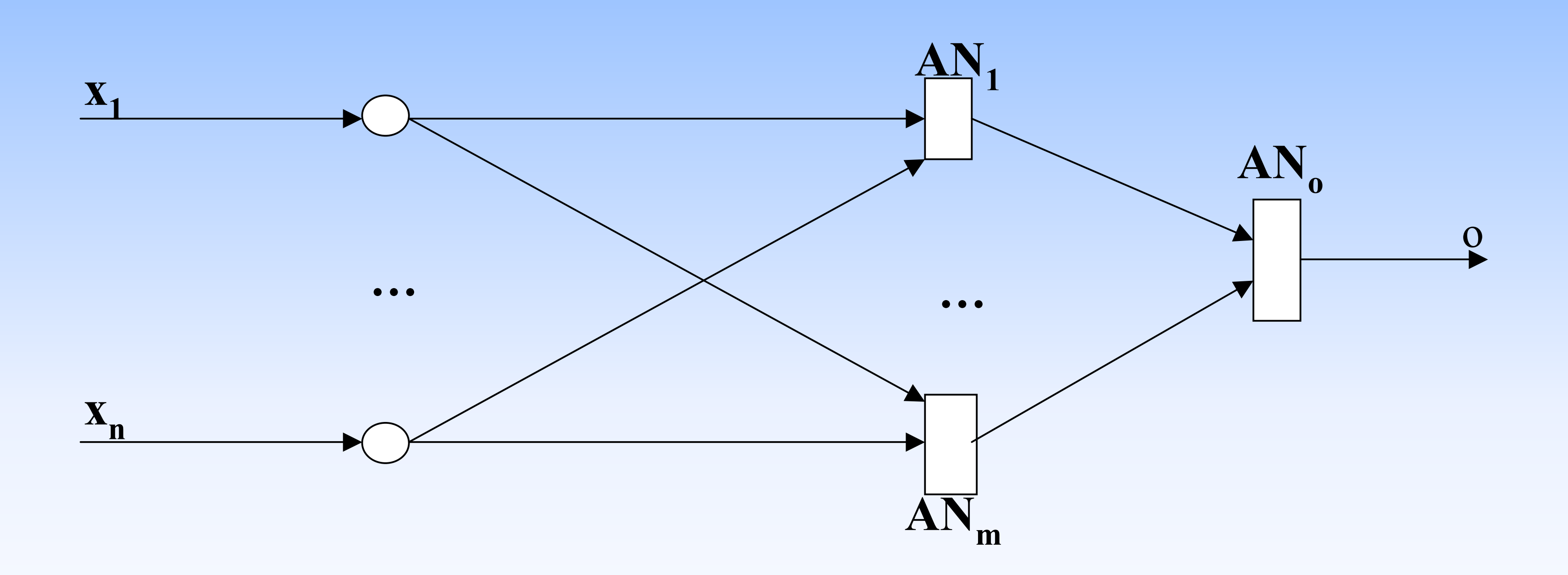
- R. O. Windner 1960年

自变量个数	函数的个数	线性可分函数的个数
1	4	4
2	16	14
3	256	104
4	65,536	1882
5	$4.3*10^{9}$	94,572
6	$1.8*10^{19}$	5,028,134

3.3.2 线性不可分问题的克服

- 用多个单级网组合在一起，并用其中的一个去综合其它单级网的结果，我们就可以构成一个两级网络，该网络可以被用来在平面上划分出一个封闭或者开放的凸域来
- 一个非凸域可以拆分成多个凸域。按照这一思路，三级网将会更一般一些，我们可以用它去识别出一些非凸域来。
- 解决好隐藏层的联接权的调整问题是非常关键的

两级单输出网在n维空间中划分出m边凸域



第4章 BP网络

- 主要内容：
  - BP网络的构成
  - 隐藏层权的调整分析
  - Delta规则理论推导
  - 算法的收敛速度及其改进讨论
  - BP网络中的几个重要问题
- 重点：BP算法
- 难点：Delta规则的理论推导

第4章 BP网络

4.1 概述

1、BP算法的出现

非循环多级网络的训练算法

UCSD PDP小组的Rumelhart、Hinton和Williams1986年独立地给出了BP算法清楚而简单的描述

1982年，Paker就完成了相似的工作

1974年，Werbos已提出了该方法

2、弱点：训练速度非常慢、局部极小点的逃离问题、算法不一定收敛。

3、优点：广泛的适应性和有效性。

4.2 基本BP算法

- **4.2.1** 网络的构成

神经元的网络输入：

$$\mathbf{net_i = x_1 w_{1i} + x_2 w_{2i} + \ldots + x_n w_{ni}}$$

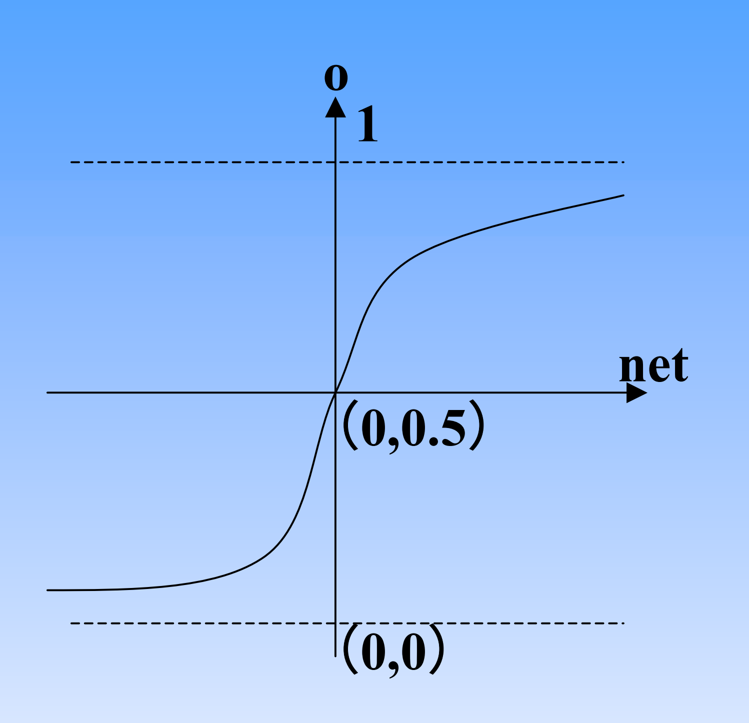
神经元的输出：

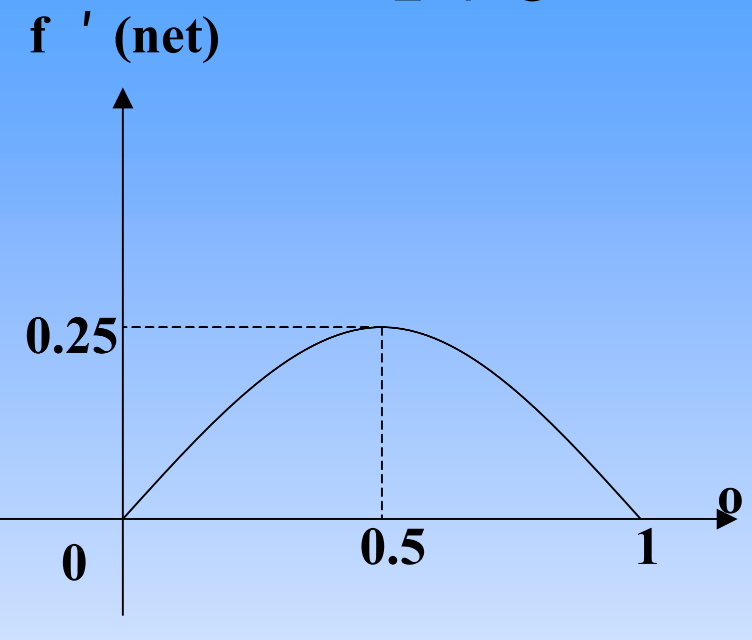
$$o = f(net) = \frac{1}{1 + e^{-net}}$$

$$f'(net) = -\frac{1}{(1+e^{-net})^2}(-e^{-net}) = o - o^2 = o(1-o)$$

输出函数分析

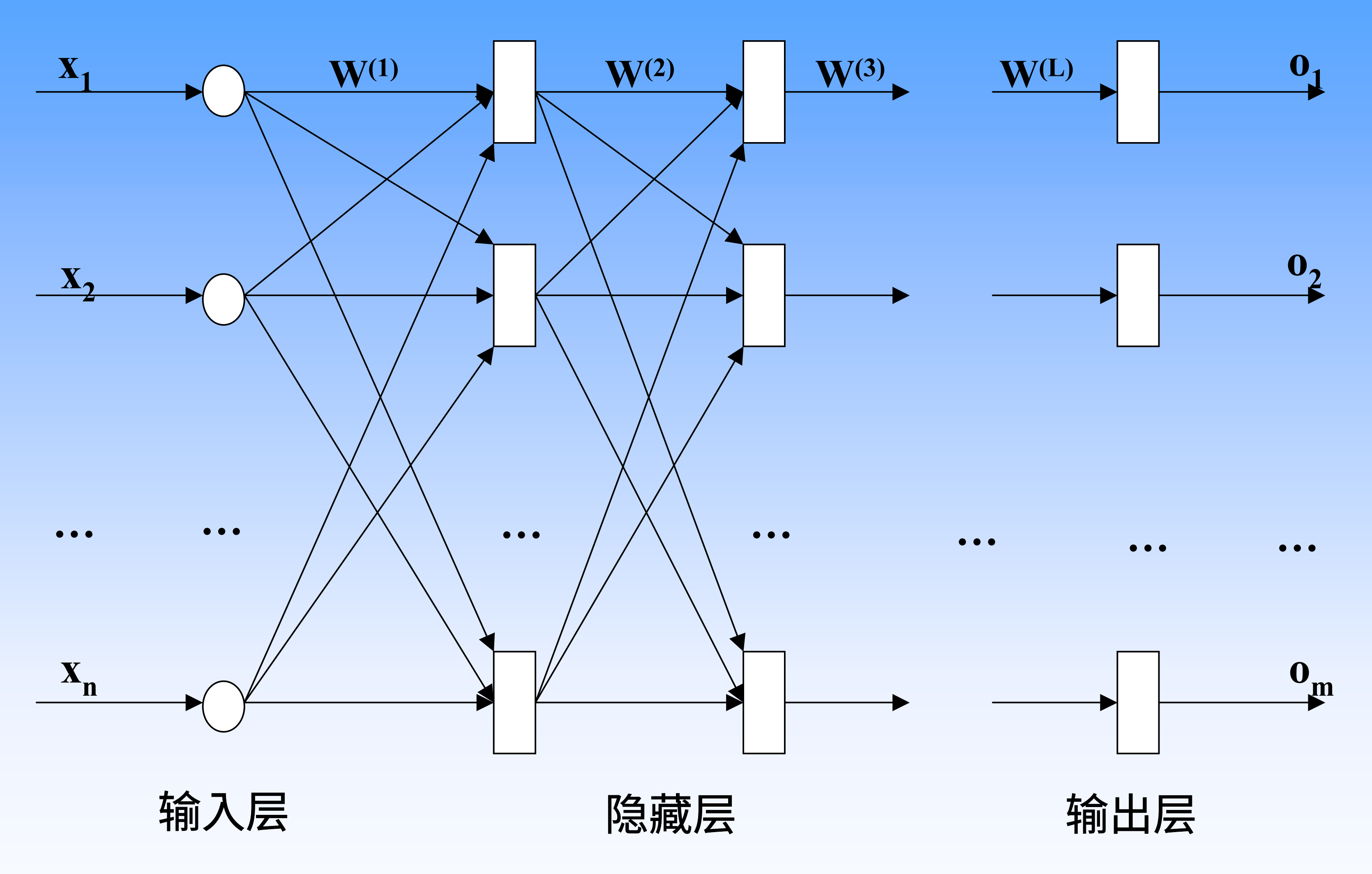
$$o=\frac{1}{1+e^{-net}}$$





– 应该将net的值尽量控制在收敛比较快的范围内
– 可以用其它的函数作为激活函数，只要该函数是处处可导的

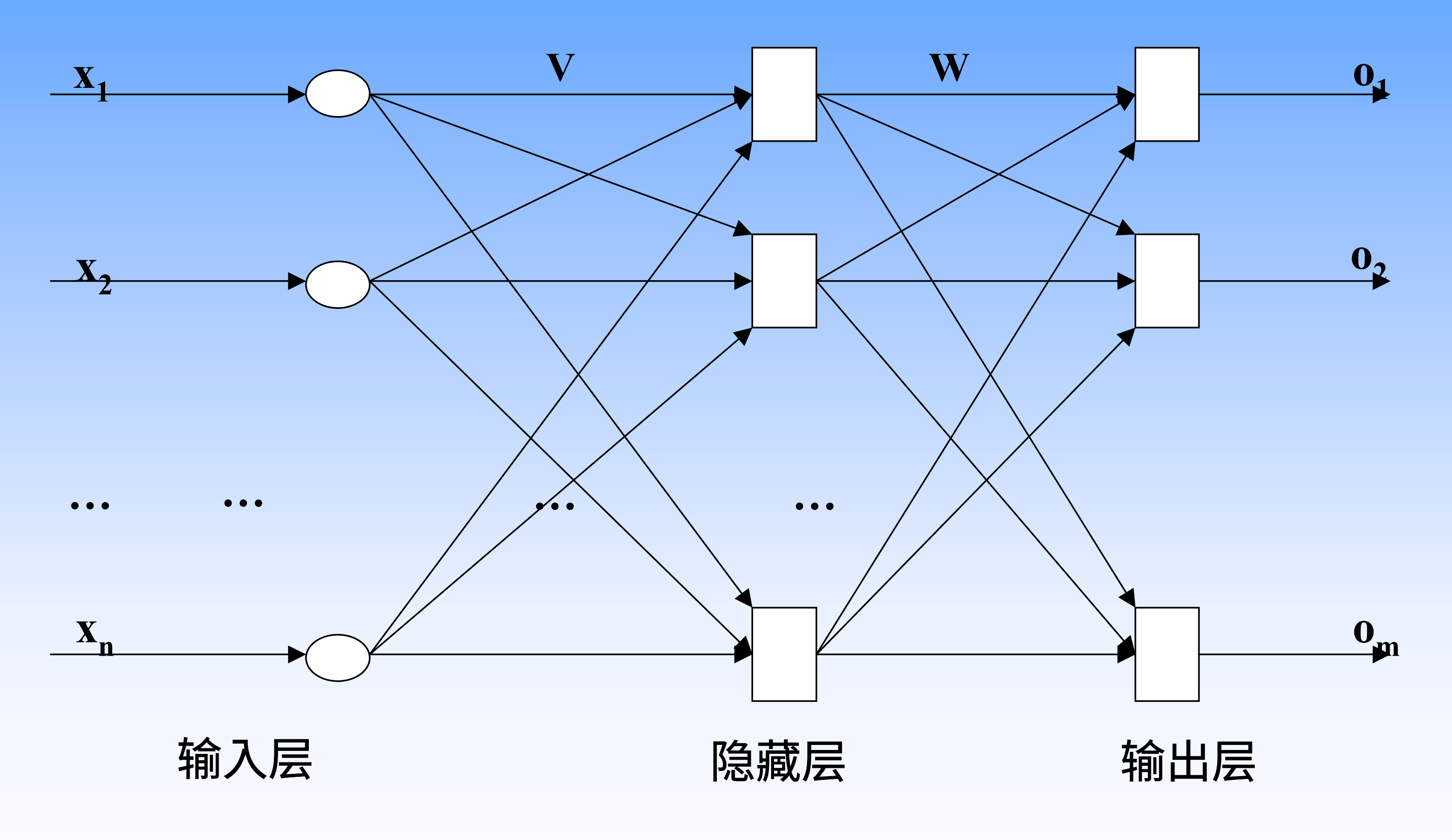
网络的拓扑结构



网络的拓扑结构

1. **BP**网的结构
2. 输入向量、输出向量的维数、网络隐藏层的层数和各个隐藏层神经元的个数的决定
3. 实验：增加隐藏层的层数和隐藏层神经元个数不一定总能够提高网络精度和表达能力。
4. BP网一般都选用二级网络。

网络的拓扑结构



4.2.2 训练过程概述

样本：(输入向量，理想输出向量)

权初始化："小随机数"与饱和状态；"不同"保证网络可以学。

1、向前传播阶段：

（1）从样本集中取一个样本(X_p，Y_p)，将X_p输入网络；

（2）计算相应的实际输出O_p：

$$O_p=F_l(\ldots(F_2(F_1(X_pW^{(1)})W^{(2)})\ldots)W^{(L)})$$

4.2.2 训练过程概述

2、向后传播阶段——误差传播阶段：

（1）计算实际输出O_p与相应的理想输出Y_p的差；

（2）按极小化误差的方式调整权矩阵。

（3）网络关于第p个样本的误差测度：

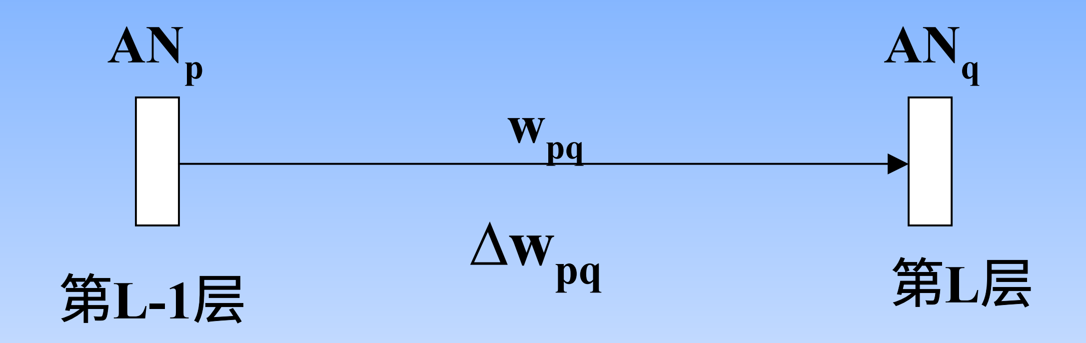
$$E_p = \frac{1}{2}\sum_{j=1}^{m}\left(y_{pj} - o_{pj}\right)^2$$

（4） 网络关于整个样本集的误差测度：

$$E = \sum_{p} E_p$$

4.2.3 误差传播分析

1、输出层权的调整



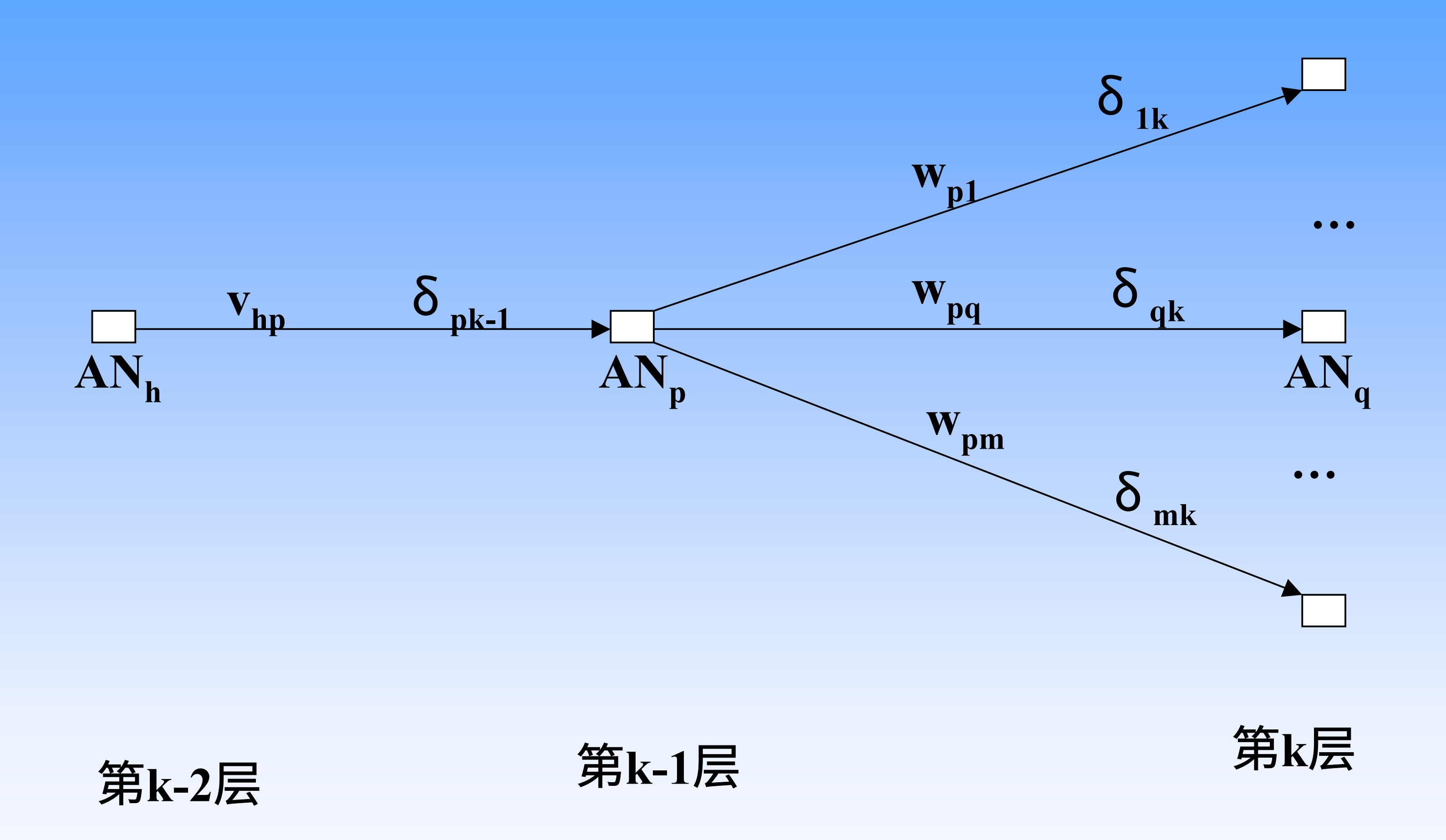
$$w_{pq} = w_{pq} + \Delta w_{pq}$$

$$\Delta w_{pq} = \alpha \delta_q o_p$$

$$= \alpha f_n' (net_q)(y_q - o_q) o_p$$

$$= \alpha o_q (1 - o_q) (y_q - o_q) o_p$$

2、隐藏层权的调整



2、隐藏层权的调整

δ_{pk-1}的值和δ_{1k}，δ_{2k}，…，δ_{mk}有关

不妨认为δ_{pk-1}

通过权w_{p1}对δ_{1k}做出贡献，

通过权w_{p2}对δ_{2k}做出贡献，

……

通过权w_{pm}对δ_{mk}做出贡献。

$$\delta_{pk-1}= f_{k-1}'(net_p)(w_{p1}\delta_{1k}+ w_{p2}\delta_{2k}+\ldots+ w_{pm}\delta_{mk})$$

2、隐藏层权的调整

$$v_{hp}=v_{hp}+\Delta v_{hp}$$

$$\Delta v_{hp}=\alpha\delta_{pk-1}o_{hk-2}$$

$$=\alpha f_{k-1}'(net_p)(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\ldots+w_{pm}\delta_{mk})o_{hk-2}$$

$$=\alpha o_{pk-1}(1-o_{pk-1})(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\ldots+w_{pm}\delta_{mk})o_{hk-2}$$

δ_{1k}

w_{p1}

…

v_{hp} δ_{pk-1} w_{pm} δ_{qk}

AN_h AN_p AN_q

w_{pq}

δ_{mk} …

第k-2层 第k-1层 第k层

4.2.4 基本的BP算法

- 样本集：$S=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_s,Y_s)\}$
- 基本思想 ：
  - 逐一地根据样本集中的样本(X_k,Y_k)计算出实际输出O_k和误差测度E_1，对$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$各做一次调整，重复这个循环，直到$\sum E_p<\varepsilon$。
  - 用输出层的误差调整输出层权矩阵，并用此误差估计输出层的直接前导层的误差，再用输出层前导层误差估计更前一层的误差。如此获得所有其它各层的误差估计，并用这些估计实现对权矩阵的修改。形成将输出端表现出的误差沿着与输入信号相反的方向逐级向输入端传递的过程

算法4-1 基本BP算法

1 for k=1 to L do

 1.1 初始化$W^{(k)}$；

2 初始化精度控制参数ε；

3 E=ε+1;

4 while E>ε do

 4.1 E=0;

算法4-1 基本BP算法

4.2 对S中的每一个样本（X_p,Y_p）：

4.2.1 计算出X_p对应的实际输出O_p；

4.2.2 计算出E_p；

4.2.3 $E=E+E_p$；

4.2.4 根据相应式子调整$W^{(L)}$；

4.2.5 $k=L-1$；

4.2.6 while $k\neq0$ do

4.2.6.1 根据相应式子调整$W^{(k)}$；

4.2.6.2 $k=k-1$

4.3 $E=E/2.0$

4.3 算法的改进

1、BP网络接受样本的顺序对训练结果有较大影响。它更“偏爱”较后出现的样本

2、给集中的样本安排一个适当的顺序，是非常困难的。

3、样本顺序影响结果的原因：“分别”、“依次”

4、用(X_1,Y_1)，(X_2,Y_2)，…，(X_s,Y_s)的“总效果”修改$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$。

$$\Delta w^{(k)}_{ij}=\sum \Delta_p w^{(k)}_{ij}$$

算法4-2 消除样本顺序影响的BP算法

1 for k=1 to L do

1.1 初始化$W^{(k)}$；

2 初始化精度控制参数ε；

3 E=ε+1;

4 while E>ε do

4.1 E=0;

4.2 对所有的i，j，k：$\Delta w^{(k)}_{ij}=0$；

4.3 对S中的每一个样本（X_p, Y_p）：

4.3.1 计算出X_p对应的实际输出O_p；

4.3.2 计算出E_p；

4.3.3 $E=E+E_p$；

4.3.4 对所有i，j根据相应式子计算$\Delta_p w^{(L)}_{ij}$；

4.3.5 对所有i，j：$\Delta w^{(L)}_{ij}=\Delta w^{(L)}_{ij}+\Delta_p w^{(L)}_{ij}$；

4.3.6 $k=L-1$；

4.3.7 while $k\neq 0$ do

4.3.7.1 对所有i, j根据相应式子计算$\Delta_p w^{(k)}_{ij}$；

4.3.7.2 对所有i, j：$\Delta w^{(k)}_{ij}=\Delta w^{(k)}_{ij}+\Delta_p w^{(k)}_{ij}$；

4.3.7.3 $k=k-1$

4.4 对所有i，j，k：$w^{(k)}_{ij}= w^{(k)}_{ij}+ \Delta w^{(k)}_{ij}$;

4.5 $E=E/2.0$

算法4-2 分析

- 较好地解决了因样本的顺序引起的精度问题和训练的抖动问题
- 收敛速度：比较慢
- 偏移量：给每一个神经元增加一个偏移量来加快收敛速度
- 冲量：联接权的本次修改要考虑上次修改的影响，以减少抖动问题

算法4-2 分析——冲量设置

- **Rumelhart等人1986年**
  - $\Delta w_{ij}=\alpha\,\delta_j o_i+\beta\,\Delta w_{ij}'$
  - $\Delta w_{ij}'$ 为上一次的修改量，β为冲量系数，一般可取到0.9
- **Sejnowski与Rosenberg，1987年**
  - $\Delta w_{ij}=\alpha((1-\beta)\delta_j o_i+\beta\,\Delta w_{ij}')$
  - $\Delta w_{ij}'$ 也是上一次的修改量，β在0和1之间取值

4.4 算法的实现

- 主要数据结构

W[H，m]——输出层的权矩阵；

V[n，H]——输入（隐藏）层的权矩阵；

Δ_o**[m]**——输出层各联接权的修改量组成的向量；

Δ_h**[H]**——隐藏层各联接权的修改量组成的向量；

O_1——隐藏层的输出向量；

O_2——输出层的输出向量；

(X，Y)——一个样本。

算法的主要实现步骤

1　用不同的小伪随机数初始化W，V；

2　初始化精度控制参数ε；学习率α ；

3　循环控制参数E=ε+1；循环最大次数M；循环次数控制参数N=0；

4　while E>ε & N<M do

4.1 N=N+1；E=0；

4.2 对每一个样本(X,Y)，执行如下操作

4.2 对每一个样本(X, Y)，执行的操作

4.2.1 计算：$O_1=F_1$（XV）；$O_2=F_2$（O_1W）；

4.2.2 计算输出层的权修改量 for i=1 to m

4.2.2.1 $\Delta_o[i]= O_2[i]*(1- O_2[i])*(Y[i]-O_2[i])$；

4.2.3 计算输出误差：for i=1 to m

4.2.3.1 $E=E+(Y[i]-O_2[i])^2$；

4.2 对每一个样本(X，Y)，执行的操作

4.2.4 计算隐藏层的权修改量：for i=1 to H

4.2.4.1 Z=0；

4.2.4.2 for j=1 to m do Z=Z+W[i, j]* Δ_o[j]；

4.2.4.3 Δ_h[i]=Z* O_1 [i](1- O_1 [i])；

4.2.5 修改输出层权矩阵：for k=1 to H & i=1 to m

4.2.5.1 W[k, i]= W[k, i]+ α*O_1[k]*Δ_o[i]；

4.2.5 修改隐藏层权矩阵：for k=1 to n & i=1 to H

4.2.5.1 V[k, i]= V[k, i]+ α*X[k]* Δ_h[i]；

建议

- 隐藏层的神经元的个数H作为一个输入参数
- 同时将ε、循环最大次数M等，作为算法的输入参数
- 在调试阶段，最外层循环内，加一层控制，以探测网络是否陷入了局部极小点

4.5 算法的理论基础

- 基本假设
  - 网络含有L层
  - 联接矩阵： $\mathbf{W}^{(1)}$ ， $\mathbf{W}^{(2)}$ ， …， $\mathbf{W}^{(L)}$
  - 第k层的神经元：$\mathbf{H}_k$个
  - 自变量数： $\mathbf{n*H_1+H_1*H_2+H_2*H_3+\ldots+H_L*m}$
  - 样本集： $\mathbf{S=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_s,Y_s)\}}$
- 误差测度:

$$E = \sum_{p=1}^{s} E_p$$

误差测度

$$E = \sum_{p=1}^{s} E_p$$

用E代表E_P，用（X，Y）代表（X_P，Y_P）

$X=(x_1, x_2, \ldots, x_n)$

$Y=(y_1, y_2, \ldots, y_m)$

该样本对应的实际输出为

$O=(o_1, o_2, \ldots, o_m)$

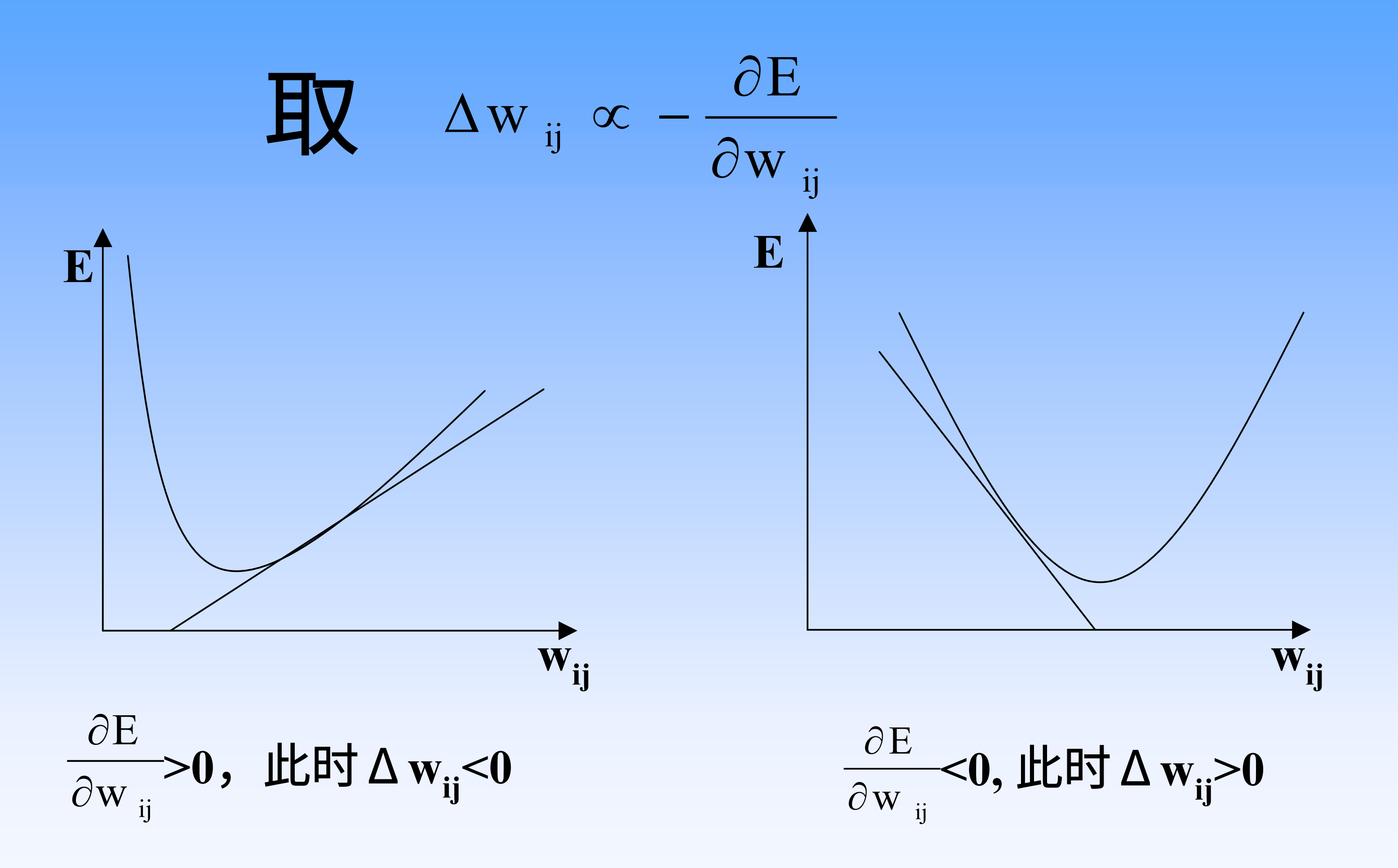
误差测度 $E = \sum_{p=1}^{s} E_p$

- 用理想输出与实际输出的方差作为相应的误差测度

$$E = \frac{1}{2} \sum_{k=1}^{m} (y_k - o_k)^2$$

最速下降法，要求E的极小点

取 $\Delta w_{ij} \propto -\dfrac{\partial E}{\partial w_{ij}}$



$\dfrac{\partial E}{\partial w_{ij}}$>0，此时Δ$w_{ij}$<0

$\dfrac{\partial E}{\partial w_{ij}}$<0, 此时Δ$w_{ij}$>0

最速下降法，要求E的极小点

$$-\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}}$$

而其中的 $\mathrm{net}_j = \sum_k \mathrm{w}_{kj}\mathrm{o}_k$

所以，$$\frac{\partial \mathrm{net}_j}{\partial \mathrm{w}_{ij}} = \frac{\partial\left(\sum_k \mathrm{w}_{kj}\mathrm{o}_k\right)}{\partial \mathrm{w}_{ij}} = \mathrm{o}_i$$

最速下降法，要求E的极小点

$$-\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}}$$

令 $\delta_j = -\frac{\partial E}{\partial net_j}$

$$= -\frac{\partial E}{\partial net_j} \cdot \frac{\partial \left(\sum_k w_{kj} o_k \right)}{\partial w_{ij}}$$

$$= -\frac{\partial E}{\partial net_j} \cdot o_i$$

所以 $\Delta \mathbf{w}_{ij} = \alpha \delta_j \mathbf{o}_i$

α为学习率

AN_j为输出层神经元

$o_j=f(net_j)$

容易得到

$$\frac{\partial o_j}{\partial net_j} = f'(net_j)$$

从而

$$\delta_j = -\frac{\partial E}{\partial net_j}$$

$$= -\frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

$$= -\frac{\partial E}{\partial o_j} \cdot f'(net_j)$$

AN_j为输出层神经元

$$-\frac{\partial E}{\partial o_j} = -\frac{\partial\left(\frac{1}{2}\sum_{k=1}^{m}(y_k - o_k)^2\right)}{\partial o_j}$$

$$= -\left(\frac{1}{2}\frac{\partial(y_j - o_j)^2}{\partial o_j}\right)$$

$$= -\left(-\frac{2}{2}(y_j - o_j)\right)$$

$$= (y_j - o_j)$$

AN_j为输出层神经元

所以，$\delta_j = (y_j - o_j) f'(net_j)$

故，当AN_j为输出层的神经元时，它对应的联接权w_{ij}应该按照下列公式进行调整：

$$w_{ij} = w_{ij} + \alpha \delta_j o_i$$

$$= w_{ij} + \alpha f'(net_j)(y_j - o_j) o_i$$

AN_j为隐藏层神经元

$$\delta_j = -\frac{\partial E}{\partial net_j}$$

$$= -\frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial o_j}{\partial net_j} = f'(net_j)$$

$$\delta_j = -\frac{\partial E}{\partial o_j} \cdot f'(net_j)$$

函数

$$E = \frac{1}{2}\sum_{k=1}^{m}(y_k - o_k)^2$$

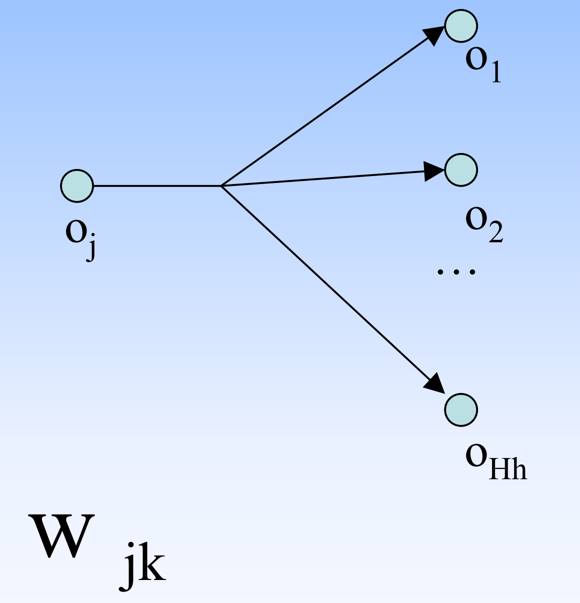
AN_j为隐藏层神经元

net_k是 o_j下一级的神经元的网络输入

$$net_k = \sum_{i=1}^{H_h} w_{ik} o_i$$

$$\frac{\partial E}{\partial o_j} = \sum_{k=1}^{H_h} \left(\frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j} \right)$$

$$\frac{\partial net_k}{\partial o_j} = \frac{\partial \left(\sum_{i=1}^{H_h} w_{ik} o_i \right)}{\partial o_j} = w_{jk}$$



AN_j为隐藏层神经元

$$\frac{\partial E}{\partial o_j} = \sum_{k=1}^{H_h} \left(\frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j} \right)$$

$$= \sum_{k=1}^{H_h} \left(\frac{\partial E}{\partial net_k} \cdot w_{jk} \right)$$

$$\delta_k = -\frac{\partial E}{\partial net_k}$$

$$\frac{\partial E}{\partial o_j} = -\sum_{k=1}^{H_h} \delta_k w_{jk}$$

AN_j为隐藏层神经元

$$\delta_j = -\frac{\partial E}{\partial o_j} \cdot f'(net_j)$$

$$= -\left(-\sum_{k=1}^{H_h} \delta_k w_{jk}\right) \cdot f'(net_j)$$

$$\delta_j = \left(\sum_{k=1}^{H_h} \delta_k w_{jk}\right) \cdot f'(net_j)$$

AN_j为隐藏层神经元

$$\Delta w_{ij} = \alpha \left(\sum_{k=1}^{H_h} \delta_k w_{jk} \right) \cdot f'(net_j) o_i$$

$$w_{ij} = w_{ij} + \alpha \left(\sum_{k=1}^{H_h} \delta_k w_{jk} \right) \cdot f'(net_j) o_i$$

4.6 几个问题的讨论

- 收敛速度问题
- 局部极小点问题
  - 逃离/避开局部极小点：修改**W**、**V**的初值——并不是总有效。
  - 逃离——统计方法；**[Wasserman，1986]**将**Cauchy**训练与**BP**算法结合起来，可以在保证训练速度不被降低的情况下，找到全局极小点。

4.6 几个问题的讨论

- 网络瘫痪问题
  - 在训练中，权可能变得很大，这会使神经元的网络输入变得很大，从而又使得其激活函数的导函数在此点上的取值很小。根据相应式子，此时的训练步长会变得非常小，进而将导致训练速度降得非常低，最终导致网络停止收敛
- 稳定性问题
  - 用修改量的综合实施权的修改
  - 连续变化的环境，它将变成无效的

4.6 几个问题的讨论

- 步长问题
  - BP网络的收敛是基于无穷小的权修改量
  - 步长太小，收敛就非常慢
  - 步长太大，可能会导致网络的瘫痪和不稳定
  - 自适应步长，使得权修改量能随着网络的训练而不断变化。[1988年，Wasserman]

第5章 对传网

- 主要内容：CPN的网络结构，正常运行，输入向量的预处理，Kohonen层的训练算法及其权矩阵的初始化方法；Grossberg层的训练；完整的对传网
- 重点：Kohonen层与Grossberg层的正常运行与训练
- 难点：Kohonen层的训练算法及其权矩阵的初始化方法

第5章 对传网

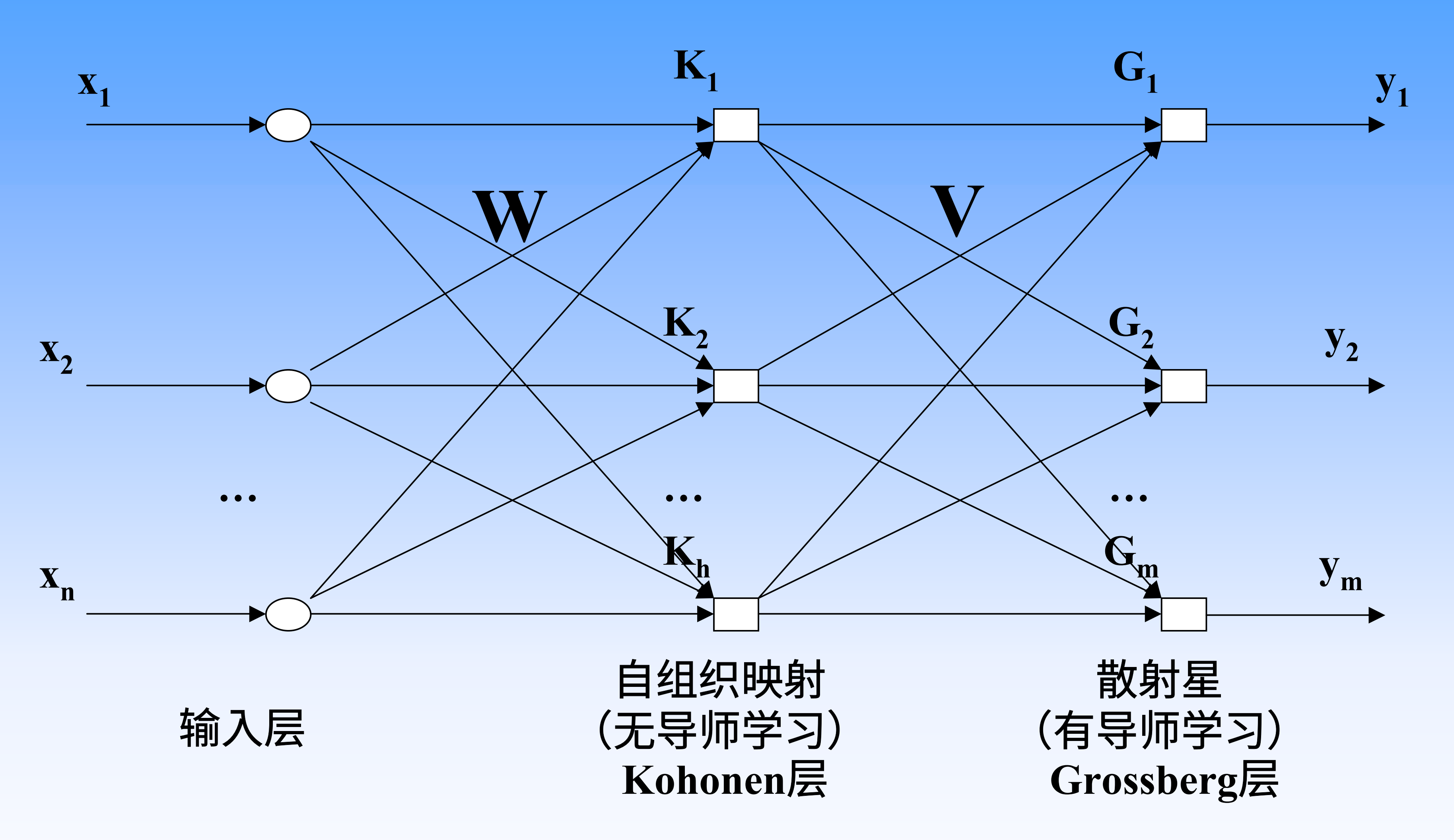
第5章 对传网

- Robert Hecht-Nielson 在1987年提出了对传网（Counterpropagation Networks，CPN）。
- CPN为异构网：
  - Kohonen1981年提出的Self-organization map
    - SOM——Kohonen层
  - Grossberg1969年提出的Outstar——Grossberg层
- 训练时间短：BP的1%。应用面：比较窄
- 让网络的隐藏层执行无导师学习，是解决多级网络训练的另一个思路

5.1 网络结构

- 单向CPN，完整CPN（双向网）
- 除拓扑结构外，网络的运行机制也是确定网络结构（同构、异构）和性能的重要因素
- *网络的层数计算*

5.1 网络结构



x1
x2
…
xn
K1
K2
…
Kh
G1
G2
…
Gm
y1
y2
ym
W
V
输入层
自组织映射
(无导师学习)
Kohonen层
散射星
(有导师学习)
Grossberg层

5.1 网络结构

- 以Kohonen层的神经元为“中心”讨论问题
- K_1
  - $W_1=(w_{11},\ w_{21},\ \ldots,\ w_{n1})^T$
  - $V_1=(v_{11},\ v_{12},\ \ldots,\ v_{1m})$
- K_2
  - $W_2=(w_{12},\ w_{22},\ \ldots,\ w_{n2})^T$
  - $V_2=(v_{21},\ v_{22},\ \ldots,\ v_{2m})$

  ……

- K_h
  - $W_h=(w_{1h},\ w_{2h},\ \ldots,\ w_{nh})^T$
  - $V_h=(v_{h1},\ v_{h2},\ \ldots,\ v_{hm})$

5.2 网络的正常运行

5.2.1 Kohonen层

- “强者占先、弱者退出”（the winner takes all）

$knet_j=XW_j$

$$=(x_1, x_2, \ldots, x_n)(w_{1j}, w_{2j}, \ldots, w_{nj})^T$$

$$=w_{1j}x_1+w_{2j}x_2+\ldots+w_{nj}x_n$$

向量形式

$KNET=(knet_1, knet_2, \ldots, knet_h)$

5.2.1 Kohonen层

- K_1，K_2，…，K_h的输出k_1，k_2，…，k_h构成向量 $K=(k_1, k_2, \ldots, k_h)$
- $1 \leqq j \leqq h$

$$k_j = \begin{cases} 1 & knet_j = Max\{knet_1, knet_2, \ldots, knet_h\} \\ 0 & \text{其它} \end{cases}$$

- 几何意义

5.2.2 Grossberg层

- Grossberg层的每个神经元G_j （1≦j≦m）

$$gnet_j = K(v_{1j}, v_{2j}, \ldots, v_{hj})^T$$
$$= (k_1, k_2, \ldots, k_h)(v_{1j}, v_{2j}, \ldots, v_{hj})^T$$
$$= k_1 v_{1j} + k_2 v_{2j} + \ldots + k_h v_{hj}$$

唯一输出1的神经元为K_o

$$gnet_j = k_1 v_{1j} + k_2 v_{2j} + \ldots + k_h v_{hj}$$
$$= v_{oj}$$

5.2.2 Grossberg层

$GNET=(gnet_1, gnet_2, \ldots, gnet_m)$

$=(v_{o1}, v_{o2}, \ldots, v_{om})$

$=V_o$

- 散射星：V_o的各个分量是从K_o到Grossberg层各神经元的联接权

5.2.2 Grossberg层

- CPN用于模式的完善，此时$n=m$：接受含有噪音的输入模式$(x_1，x_2，…，x_n)$，而输出去掉噪音后的模式$(v_{o1}，v_{o2}，…，v_{om})$
- 对训练启示
  - $W_1，W_2，…，W_h$，各类X的共同特征
  - $V_1，V_2，…，V_h$，X对应的理想输出Y的共同特征

5.3 Kohonen层的训练

5.3.1 输入向量的预处理

单位化处理

$$X=(x_1, x_2, \ldots, x_n)$$

$$X'=(x_1', x_2', \ldots, x_n')$$

$$=(x_1/\|X\|, x_2/\|X\|, \ldots, x_n/\|X\|)$$

5.3.2 训练

算法 5-1 Kohonen层训练算法

1　对所有的输入向量，进行单位化处理；

2　对每个样本（X，Y）执行下列过程

2.1 for j=1 to h do 根据相应式子计算**$knet_j$**；

2.2 求出最大的$knet_o$:

2.2.1 $max=knet_1$；o=1

2.2.2 for j=1 to h do

if $knet_j$>max then {max=$knet_j$；o=j}；

算法 5-1 Kohonen层训练算法

2.3 计算K

  2.3.1 for j=1 to h do k_j=0；

  2.3.2 k_o=1；

2.4 使W_o更接近X：$W_o^{(new)}=W_o^{(old)}+\alpha(X-W_o^{(old)})$;

2.5 对$W_o^{(new)}$进行单位化处理

$$W_o^{(new)}=W_o^{(old)}+\alpha(X- W_o^{(old)})$$

$\alpha \in (0, 1)$

$$W_o^{(new)}=W_o^{(old)}+\alpha(X- W_o^{(old)})$$

$$= W_o^{(old)}+\alpha X-\alpha W_o^{(old)}$$

$$X-W_o^{(new)}=X-[W_o^{(old)}+\alpha(X- W_o^{(old)})]$$

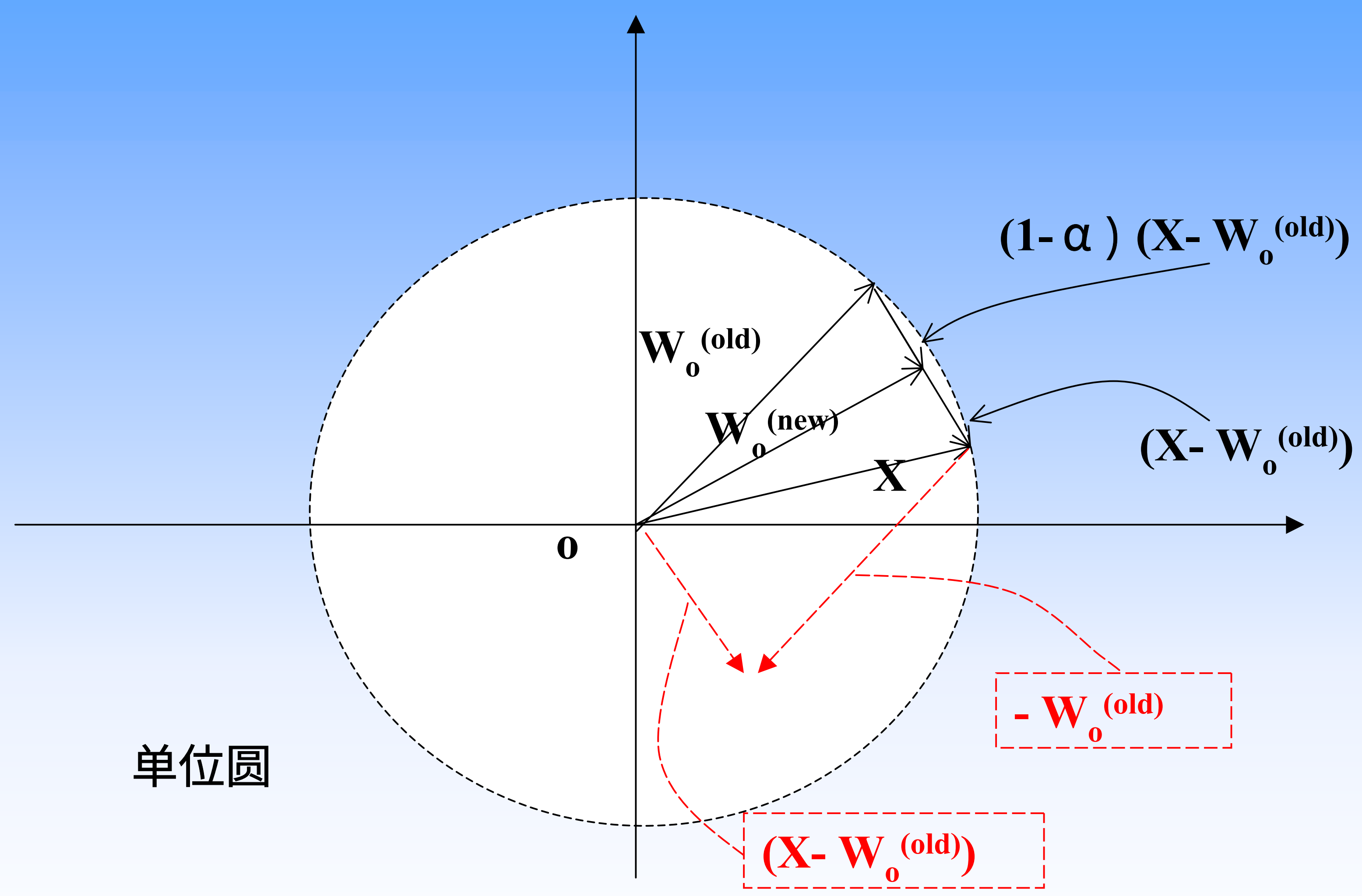
$$=X-W_o^{(old)}-\alpha X+\alpha W_o^{(old)}$$

$$= X(1-\alpha) -W_o^{(old)}(1-\alpha)$$

$$=(1-\alpha)(X-W_o^{(old)})$$

由$0<(1-\alpha)<1$，$W_o^{(new)}$比$W_o^{(old)}$更接近X

$$W_o^{(new)} = W_o^{(old)} + \alpha (X - W_o^{(old)})$$



学习率α

- 训练初期，α一般取0.7左右，它将随着训练进展不断变小
- α过大可能导致有的X被放入错误的类中；使训练陷入抖动
- 根据X的分布决定W的初值：防止类过小和过大

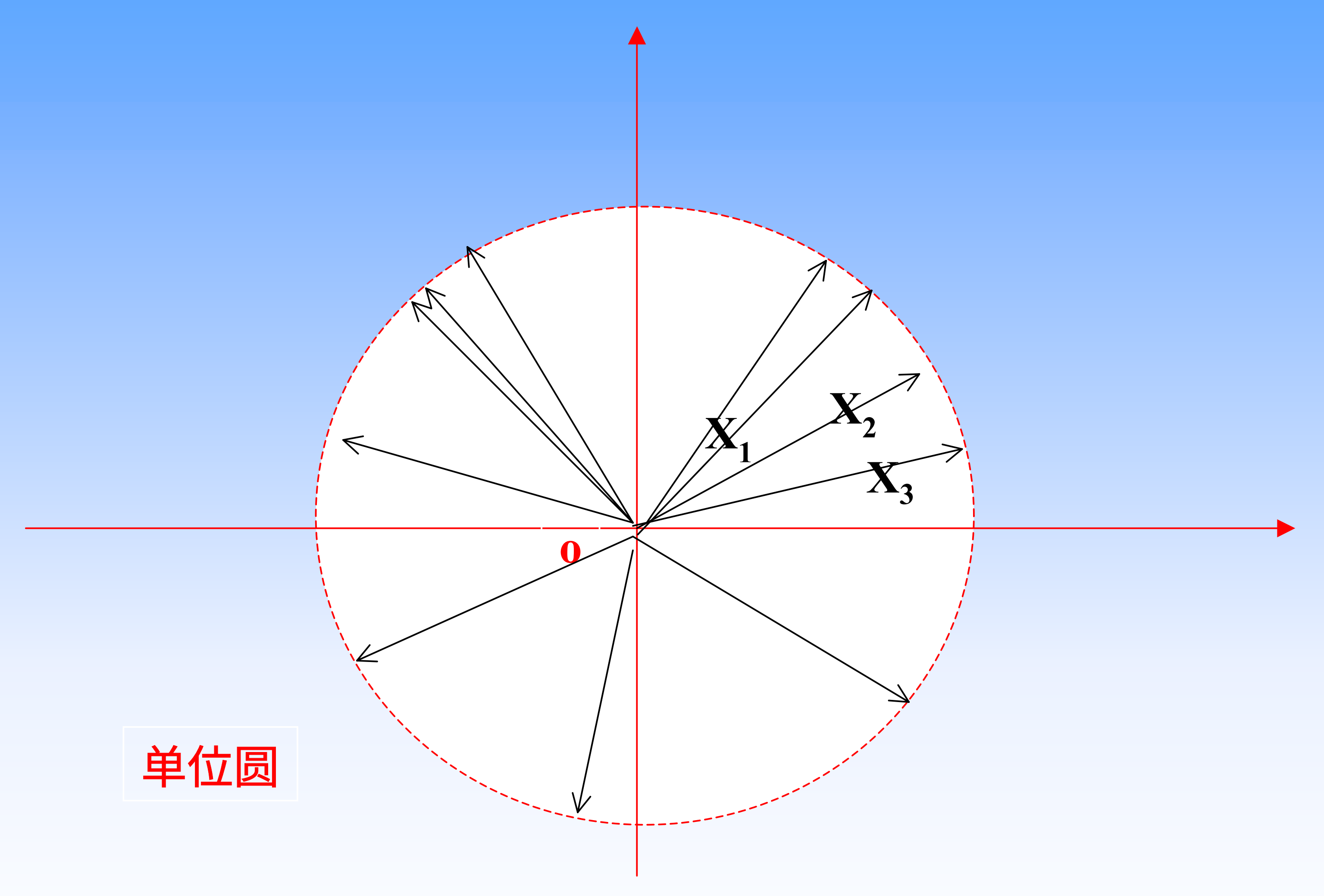
启发

- 一般来说，一个类含有许多向量。这个类对应的W_j应该是样本集中这一类向量（输入向量部分）的平均值。
- 事先给问题一个粗略分类，并从这个分类中提取一个较有代表性的向量构成样本集
- 启发我们采用训练和直接设定权向量的方式来完成该层的训练。

5.4 Kohonen层联接权初始化

- 理想情况下，W_1，W_2，…，W_h的初值应该依照样本集中的输入向量的分布来确定
- 样本集中的输入向量的分布并不是均匀的

X_i的非均匀分布要求W_i非均匀分布



凸状组合法

取 $\mathbf{w}_{\mathbf{ij}} = \frac{1}{\mathrm{sqrt}(n)}$

将输入向量

$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

变换为

$\mathbf{X}' = (\mathbf{x}_1', \mathbf{x}_2', \ldots, \mathbf{x}_n')$

其中

$$x_j' = \lambda x_j + \frac{1-\lambda}{\sqrt{n}}$$

凸状组合法

在训练的初期阶段，λ的值非常小，使得

$$X \approx \left(\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}\right)$$

随着训练的进行，λ趋近于1，从而使X'趋近于X，进而W_j趋近于一组X的平均值。

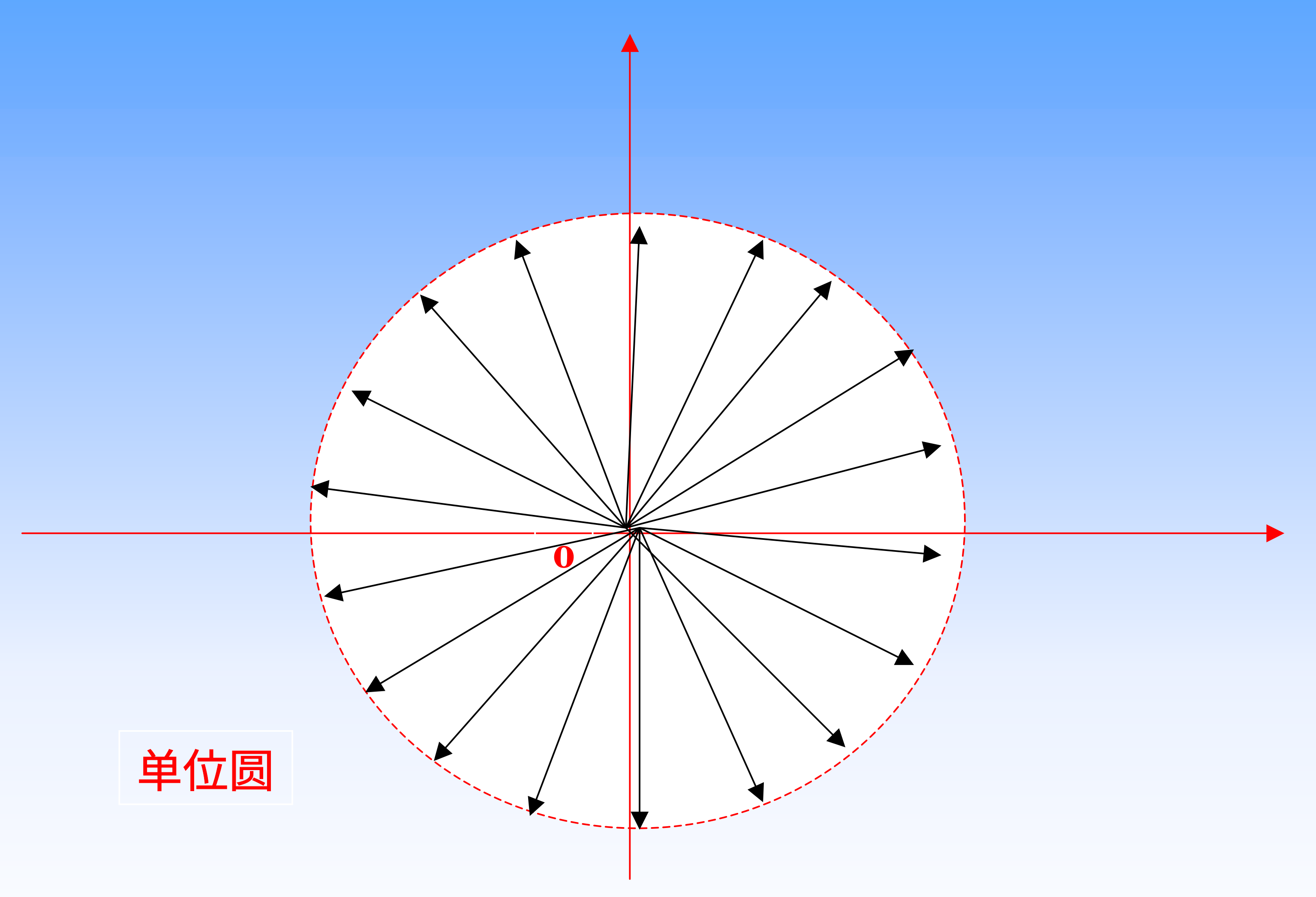
W需要追踪一个变化的目标

添加噪音法

- 在输入向量中加进适当的随机噪音，使输入向量的分布均匀。训练中逐渐去掉噪音
- W_j不断地调整自己的“运动方向”，去追踪其不断变化的目标。试验表明，这种方法的收敛速度比凸状组合法更慢。

W也需要追踪一个变化的目标

X在加噪音后变成均匀分布的



初期全调法

- **Kohonen**层训练的初期，对应一个输入向量，允许多个神经元同时处于激发状态。逐渐减少被激发的神经元的最大个数或者逐渐提高阈值，最后达到对一个输入向量，只有一个神经元激发
- 要解决的问题
  - 问题调整的范围的度量。

初期全调法

- 另一种实现
  - 在训练的初期，算法不仅调整“获胜”的神经元对应的权向量，而且对其它的权向量也作适当的调整。随着训练的推进，被调整的范围逐渐缩小，直到最终只有“获胜”的神经元对应的权向量才被调整
- 要解决的问题
  - 问题调整的范围的度量。
  - 其它的权向量的“适当调整”

DeSieno法

- 当某一个权向量所获得的匹配向量超过给定的数（1/h）后，它的阈值就被临时提高
- 问题：当最应该被某个神经元对应的权向量匹配的输入向量在较后的时候被输入时，它可能被拒绝，从而造成网络精度的损失
- **Kohonen [1988]**：在一个被完全训练过的网中，随机选取的输入向量与任何给定权向量是最接近的概率是1/h
  - 按均匀分布初始化的权向量具有相同被匹配概率

5.5 Grossberg层的训练

- 训练
  - 标量形式

$$v_{oj}= v_{oj}+\alpha(y_j - v_{oj})$$

  - 向量形式

$$V_o^{(new)}= V_o^{(old)}+\alpha(Y- V_o^{(old)})$$

- 比较

$$W_o^{(new)}=W_o^{(old)}+\alpha(X- W_o^{(old)})$$

Kohonen层

算法5-2 CPN训练算法一

0 对W、V进行初始化；

1 对所有的输入向量，进行单位化处理；

2 对每个样本（X，Y）执行下列过程

2.1 for j=1 to h do 根据$knet_j=XW_j$计算$knet_j$；

2.2 求出最大的$knet_o$：

2.2.1 $max=knet_1$；o=1;

2.2.2 for j=1 to h do

2.2.2.1 if $knet_j>max$ then {$max=knet_j$；o=j}；

算法5-2 CPN训练算法一

2.3 计算K：

  2.3.1 for j=1 to h do k_j=0；

  2.3.2 k_o=1；

2.4 使W_o更接近X：

  $W_o^{(new)}=W_o^{(old)}+\alpha(X-W_o^{(old)})$；

2.5 对$W_o^{(new)}$进行单位化处理；

2.6 使V_o更接近Y：

  $V_o^{(new)}=V_o^{(old)}+\alpha(Y-V_o^{(old)})$。

算法5-3 CPN训练算法二

- 对应Kohonen的每一个K_i，它将代表一组输入向量，所以希望这个K_i对应的V_i能代表这组输入向量对应的输出向量的平均值。

0　对W、V进行初始化；

0′清空Kohonen层各神经元对应的纪录表：

for j=1 to h do $SK_j=\Phi$；

1 对所有的输入向量，进行单位化处理；

算法5-3 CPN训练算法二

2 对每个样本（X_s，Y_s）执行下列过程

2.1 for j=1 to h do

2.1.1 根据相应式子计算$knet_j$；

2.2 求出最大的$knet_o$：

2.2.1 max=$knet_1$；o=1;

2.2.2 for j=1 to h do

2.2.2.1 if $knet_j$>max then {max=$knet_j$；o=j}；

算法5-3 CPN训练算法二

2.3 计算K：

2.3.1 for j=1 to h do k_j=0；

2.3.2 k_o=1；

2.4 使W_o更接近X_s：

$W_o^{(new)}=W_o^{(old)}+\alpha(X_s-W_o^{(old)})$；

2.5 对$W_o^{(new)}$进行单位化处理；

2.6 将Y_s放入SK_o：

$SK_o=SK_o \cup \{Y_s\}$；

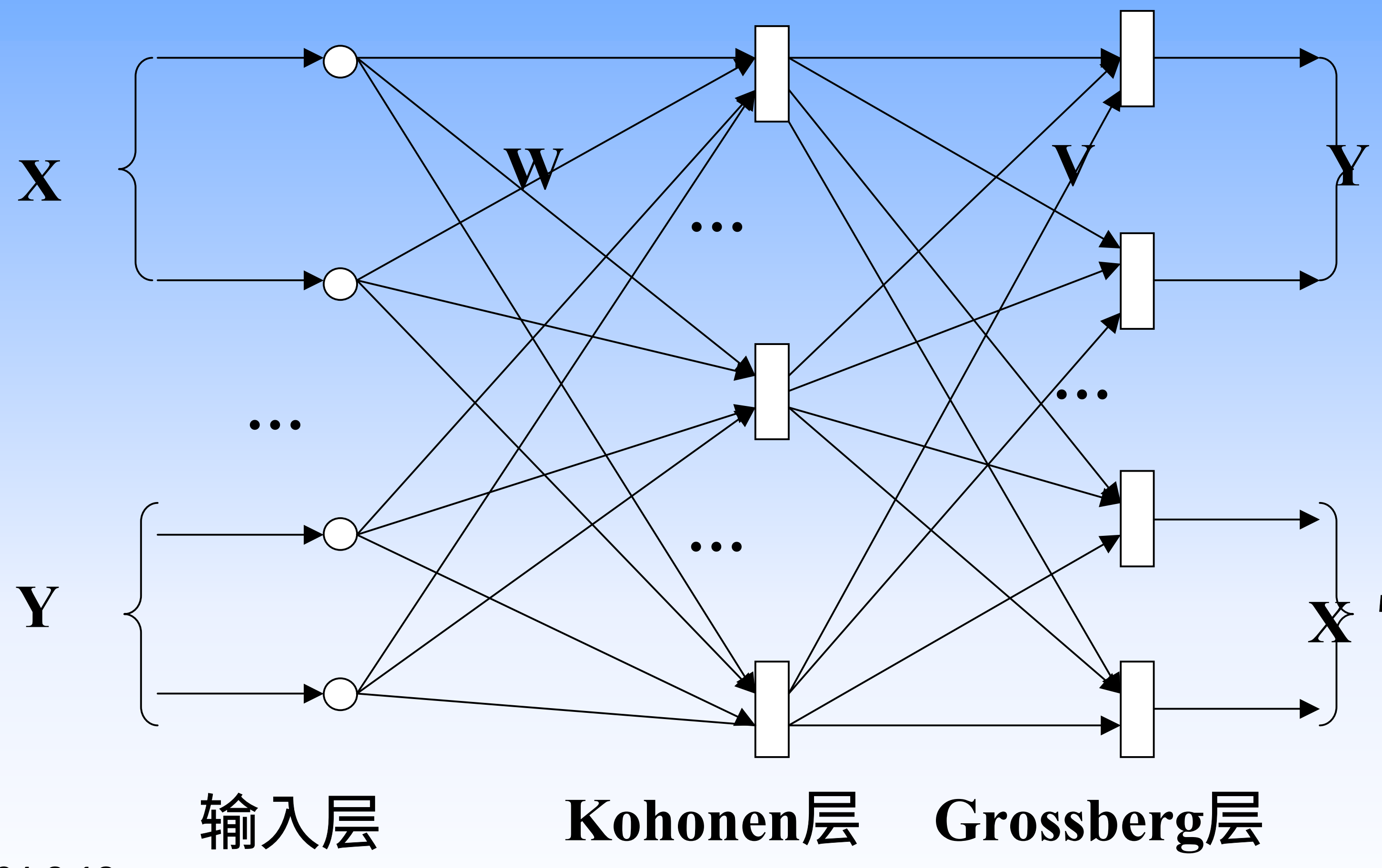
3 for j=1 to h do

V_j= SK_j中各向量的平均值

算法的进一步优化

- 集合变量SK_1， SK_2 ，…，SK_h改为其它存储量更小，而且更容易实现的变量
- 在X_s激发K_o时，Y_s被放入到SK_o中
  - 会不会出现一个向量被放入多个SK中的问题
  - 如何解决

5.6 补充说明
1、全对传网



2、非简单工作方式

- 对给定的输入向量，**Kohonen**层各神经元可以给出不同的输出
- 输出作为修改因子
  - 对应神经元**Kohonen**层、**Grossberg**层的权向量
  - 输出值较大的，表明该输入向量与该神经元对应的类较接近，它对应的权向量的修改量就大
  - 输出值较小的，表明该输入向量与该神经元对应的类较远，它对应的权向量的修改量就小。

第6章 非确定方法

- 主要内容：
  - 统计网络的基本训练算法
  - 模拟退火算法与收敛分析
  - Cauchy训练
  - 人工热与临界温度在训练中的使用
  - BP算法与Cauchy训练的结合。
- 重点：统计网络的基本训练算法，BP算法与Cauchy训练的结合
- 难点：模拟退火算法与收敛分析

第6章 非确定方法

6.1 基本的非确定训练算法

6.2 模拟退火算法

6.3 Cauchy训练

6.4 相关的几个问题

第6章 非确定方法

- 确定的方法
  - 前几章所给方法的共同特征
- 非确定的方法
  - 生物神经网络按照概率运行
- 别称
  - 统计方法（Statistical Method）。
- 既可以用于训练，又可以用于运行

6.1 基本的非确定训练算法

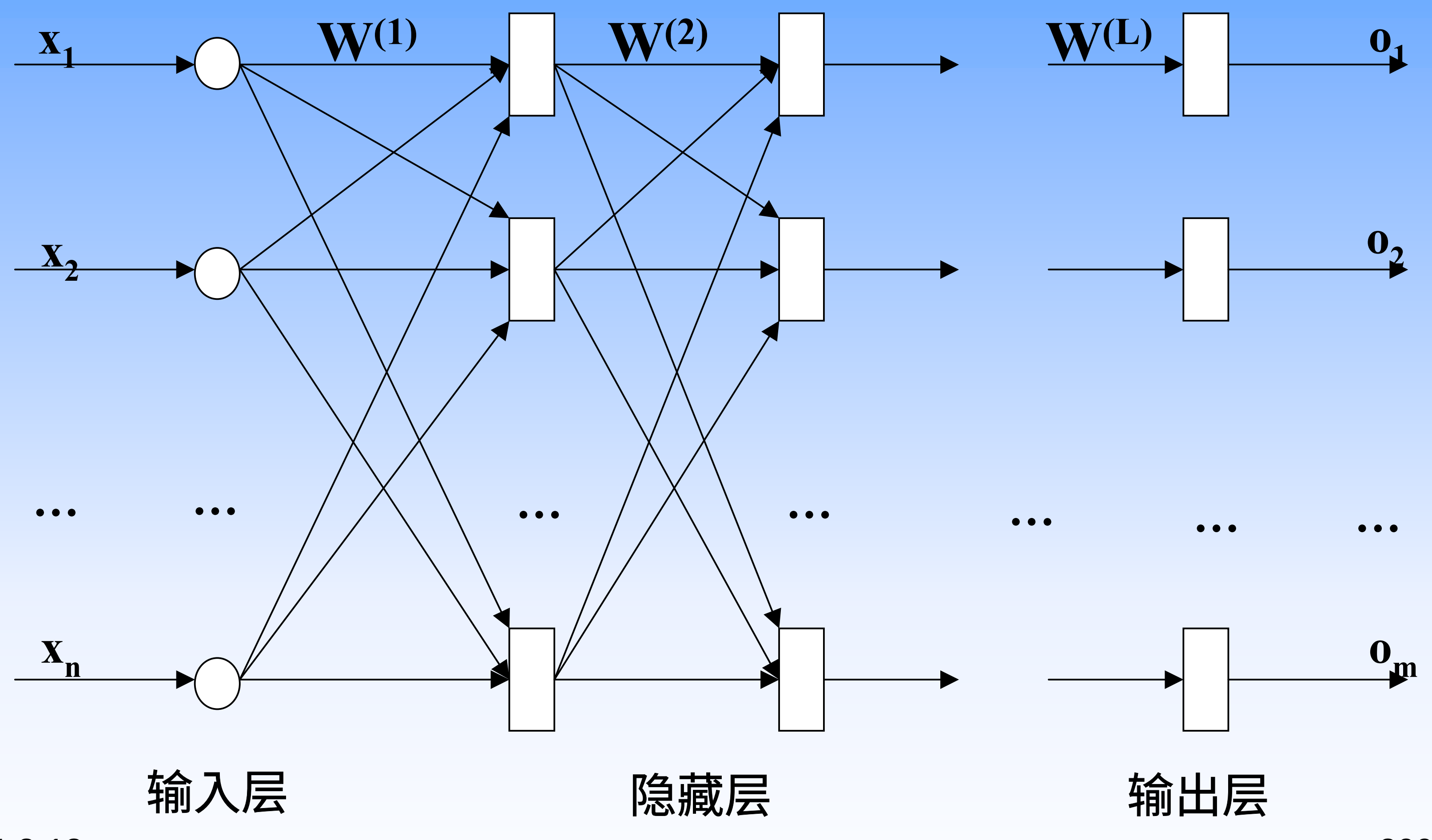
- 基本思想
  - 从所给的网络中“随机地选取一个联接权”，对该联接权提出一个“伪随机调整量”，当用此调整量对所选的联接权进行修改后，如果“被认为”修改改进了网络的性能，则保留此调整；否则放弃本次调整。

6.1 基本的非确定训练算法

- 基本数据结构
  - 样本集：$S=\{(X_1,Y_1),(X_2,Y_2),\ldots,(X_s,Y_s)\}$
  - 输入向量：$X=(x_1, x_2, \ldots, x_n)$
  - 理想输出向量：$Y=(y_1, y_2, \ldots, y_m)$
  - L层： $W^{(1)}, W^{(2)}, \ldots, W^{(L)}$

6.1 基本的非确定训练算法

- 拓扑结构



算法6-1 基本统计训练算法

1 从样本集S中取一样本（X,Y）；

2 将X输入到网络中，计算出实际输出O；

3 求出网络关于Y，O的误差测度E；

4 随机地从$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$中选择一个联接权$w_{ij}^{(p)}$；

5 生成一个小随机数$\Delta w_{ij}^{(p)}$；

6 用$\Delta w_{ij}^{(p)}$修改$w_{ij}^{(p)}$；

算法6-1 基本统计训练算法

7 用修改后的$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$重新计算X对应的实际输出O′；

8 求出网络关于Y，O′的误差测度E′；

9 如果E′<E，则保留本次对$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$的修改，

否则，根据概率判断本次修改是否有用，如果认为有用，则保留本次对$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$的修改，如果认为本次修改无用，则放弃它；

10 重复上述过程，直到网络满足要求。

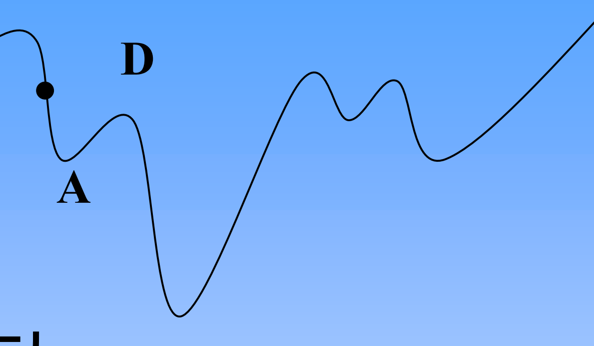
算法6-1 基本统计训练算法

- 目标函数（Objective Function）
  - 误差测度函数：实际输出与理想输出方差和
- 计算量
  - 从$W^{(1)}$，$W^{(2)}$，…，$W^{(L)}$中随机地选择w_{ij}
  - 共有$n\times H_1+H_1\times H_2+H_2\times H_3+\ldots+H_{M-1}\times m$个“变量”可供选择
- 伪随机数
  - 伪随机数发生器来产生$\Delta w_{ij}^{(p)}$；
  - 按照所谓的“能量”函数的分布去计算它

算法6-1 基本统计训练算法

- 局部极小点
  - 当E′<E不成立时，考虑使网络从局部极小点中逃离出来，必须允许目标函数暂时变坏
- 循环控制
  - 判断标准
  - 用一个样本对网络的某一个联接权进行修改后，是随机地抽取另一个联接权进行重复，还是再选择下一个样本进行重复
  - 对一个选定的样本，每次是否可以选取若干个联接权进行修改？如果可以，还应做什么工作？

逃离局部极小点

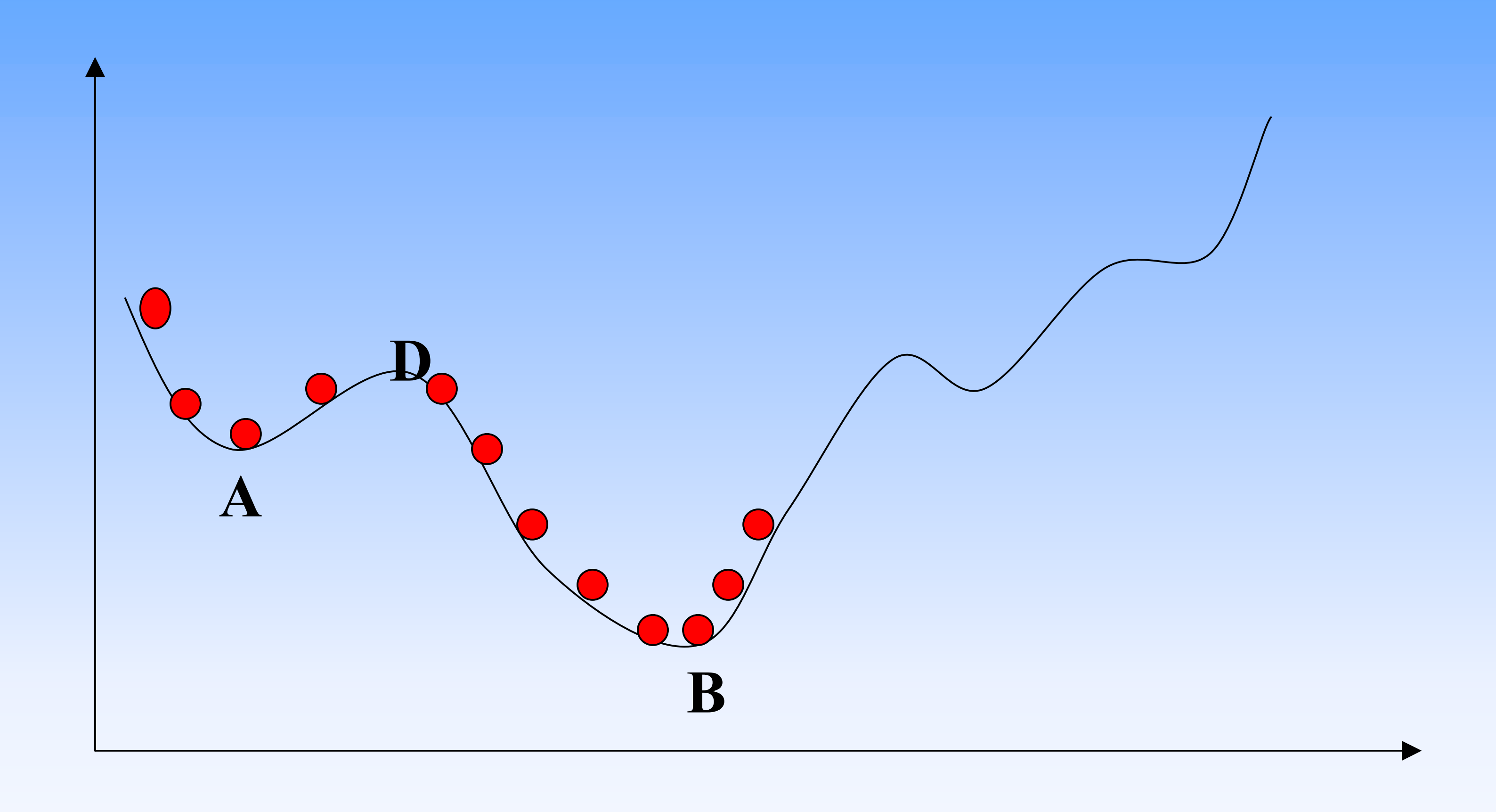


- 联接权修改量
  - 太小：落到A点后很难逃离
  - 太大：导致在A、B两点来回抖动



- 解决办法
  - 控制联接权修改量的大小：权修改量由大变小
  - 允许暂时变坏
- 修改量的大小和网络的“能量”相关
  - 模拟退火

逃离局部极小点



6.2 模拟退火算法

- 金属中原子的能量与温度有关
- 原子能量高的时候，有能力摆脱其原来的能量状态而最后达到一个更加稳定的状态——全局极小能量状态
- 在金属的退火过程中，能量的状态分布

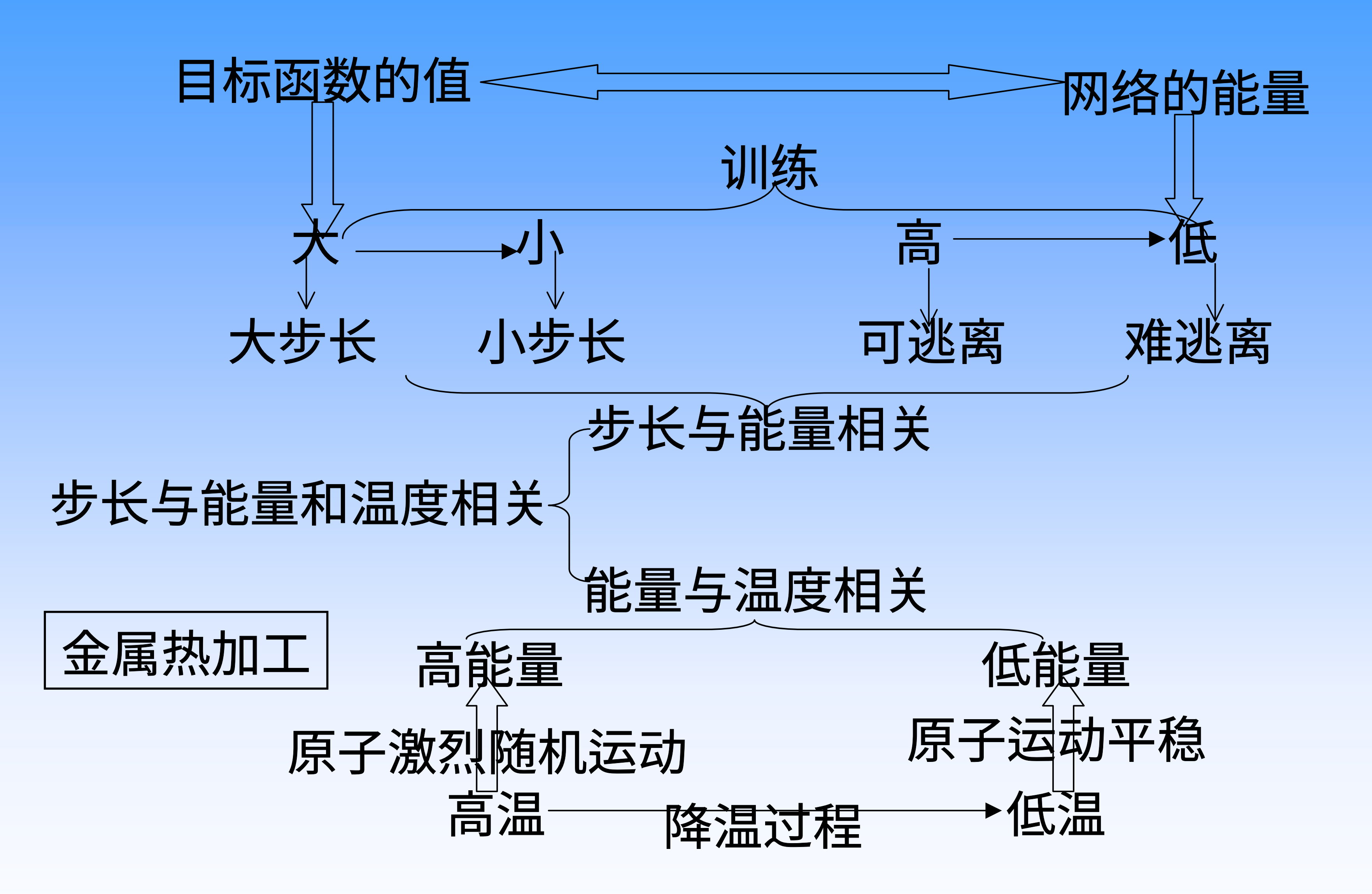
$$P(E) \propto \exp\left(-\frac{E}{kT}\right)$$

P(E)——系统处于具有能量**E**的状态的概率；

k——**Boltzmann**常数；

T——系统的绝对温度(**Kelvin**)

步长和能量、温度的关系


目标函数的值
网络的能量
训练
大
小
高
低
大步长
小步长
可逃离
难逃离
步长与能量相关
步长与能量和温度相关
能量与温度相关
金属热加工
高能量
低能量
原子激烈随机运动
原子运动平稳
高温
降温过程
低温

能量与温度

$$\lim_{T\to\infty}\left(\exp(-\frac{E}{kT})\right)=1$$

高温情况下：

T足够大，对系统所能处的任意能量状态**E**，有

$$\exp\left(-\frac{\mathrm{E}}{\mathrm{kT}}\right)$$ 将趋近于**1**

能量与温度

中温情况下：

T比较小，E的大小对P(E)有较大的影响，设$E_1>E_2$

$P(E_2)>P(E_1)$。即，系统处于高能量状态的可能性小于处于低能量状态的可能性

能量与温度

$$\lim_{T\to 0}\frac{P(E_1)}{P(E_2)} = \lim_{T\to 0}\left(\frac{\exp(-\frac{E_1}{kT})}{\exp(-\frac{E_2}{kT})}\right)$$

$$= \lim_{T\to 0}\left(\exp(-(\frac{E_1}{kT}-\frac{E_2}{kT}))\right)$$

$$= \lim_{T\to 0}\left(\exp(-\frac{E_1-E_2}{kT})\right)$$

$$= \lim_{T\to 0}\left((\frac{1}{\exp(E_1-E_2)})^{\frac{1}{kT}}\right)$$

$$= 0$$

能量与温度

低温情况下：

T非常小，E的大小对P(E) 的影响非常大，

设$E_1>E_2$

$P(E_2) >> P(E_1)$。即，当温度趋近于0时，

系统几乎不可能处于高能量状态

模拟退火组合优化法

- 目标函数——能量函数
- 人工温度T——一个初值较大的数
- 依据网络的能量和温度来决定联接权的调整量（称为步长）。
- 与金属的退火过程（Annealing）非常相似

模拟退火组合优化法

- 基本思想
  - 随机地为系统选择一个初始状态$\{w_{ij}^{(p)}\}$，在此初始状态下，给系统一个小的随机扰动$\Delta w_{ij}^{(p)}$，计算系统的能量变化
  - $\Delta E=E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})-E(\{w_{ij}^{(p)}\})$
  - 若 $\Delta E<0$ 则接受
  - 若$\Delta E\geq 0$ 则依据概率$\exp\left(-\frac{\Delta E}{kT}\right)$判断是否被接受
  - 若接受，则系统从状态$\{w_{ij}^{(p)}\}$变换到状态$\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}$；否则，系统保持不变

模拟退火组合优化法

– 在这个过程中，逐渐地降低温度T。所得的系统状态序列$\{w_{ij}^{(p)}\}$将满足下列分布

$$f = c(T)\exp\left(-\frac{E(\{w_{ij}^{(p)}\})}{kT}\right)$$

$$c(T) = \frac{1}{\sum \exp\left(-\frac{E(\{w_{ij}^{(p)}\})}{kT}\right)}$$

算法6-2 模拟退火算法

1初始化个层的联接权矩阵W；定义人工温度T的初值；

2 对每一个温度T重复如下过程：

2.1 取一样本，计算其输出与目标函数$E(\{w_{ij}^{(p)}\})$；

2.2 随机地从$\{w_{ij}^{(p)}\}$中选取一个$w_{ij}^{(p)}$；

2.3 按一定的算法产生$w_{ij}^{(p)}$的一个调整量$\Delta w_{ij}^{(p)}$；

2.4 按照$\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}$重新计算相应输出和目标函数$E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})$；

2.5 $\Delta E= E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})- E(\{w_{ij}^{(p)}\})$；

算法6-2 模拟退火算法

2.6　if ΔE>0 then

2.6.1　按均匀分布在[0,1]区间取一随机数r；

2.6.2　按Boltzmann分布计算接受本次调整的概率：

$$P(E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}))=\exp\left(-\frac{E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\})}{kT}\right)$$

2.6.3　if　$P(E(\{w_{ij}^{(p)}+\Delta w_{ij}^{(p)}\}))<r$　then 转2.2；

算法6-2 模拟退火算法

2.7 用{ $w_{ij}^{(p)} + \Delta w_{ij}^{(p)}$ }代替{ $w_{ij}^{(p)}$ }；

2.8 if 样本集中还有未被选用的样本 then 转 2.1；

3 判断在此温度下，检验Metropolis抽样是否稳定。如不稳定，则直接转2；

4 降低温度T；

5 如果T足够小，则结束，否则，转2。

算法6-2 模拟退火算法

- 算法的第2步原则上应该对每一个样本调整每一个权，调整的顺序是随机的；
- 温度T的降低
  - T=λT
  - λ叫做冷却率，一般情况下可以在[0.8，0.9]之间取值
  - Geman(1984年)：温度下降必须与时间的对数成反比，网络最终才能收敛到全局极小点

$$T = \frac{T_0}{\log(1+t)}$$

算法6-2 模拟退火算法

- T的初值T_0
  - T_0= E({$w^{(h)}$ })；即：取初始系统目标函数（能量）的值
  - T_0=z E({$w^{(h)}$ })。即：取初始系统目标函数（能量）值的若干倍
  - 按照经验给出

算法6-2 模拟退火算法

- 调整量$\Delta w_{ij}^{(p)}$的计算
  - 可以根据Boltzmann分布或者Gaussian分布来计算。也可以用其它的方法。下面讨论按Gaussian分布进行计算的方法。我们取如下形式的Gaussian分布函数。简洁起见，用符号w代替符号$w_{ij}^{(p)}$：

$$p(\Delta w)=\exp\left(-\frac{\Delta w^2}{T^2}\right)$$

Monte Carlo法

- 数值积分法
  - 根据网络的精度要求，设一个积分步长δ，然后通过数值积分构造出如下形式的表格

Δ w	δ	2δ	3δ	4δ	…	Nδ
$\int_0^{\Delta w} p(x)dx$	C_1	C_2	C_3	C_4	…	C_N

Monte Carlo法

首先按照均匀分布在[C_1，C_N]中随机地取一个值C，然后，从

$$\{C_1, C_2, C_3, \ldots, C_N\}$$

中选取C_k满足：

$$|C_k - C| = \min\{|C - C_1|, |C - C_2|, |C - C_3|, \ldots, |C - C_N|\}$$

C_k对应的kδ就是所需要的联接权调整量Δw

6.3 Cauchy训练

- **Boltzmann分布**
- **Boltzmann训练**
- **1987年，S. Szu和R. Hartley提出用Cauchy分布去取代Gaussian分布**

Cauchy分布 p(x)= $\frac{T}{T^2 + x^2}$

6.3 Cauchy训练——优点

- 对于[C_1，C_N]中的任意一个C，它按照Cauchy分布所能取到的联接权的调整量要大于按照Boltzmann分布所能取到的联接权的调整量
- 用Cauchy分布取代Boltzmann分布后，温度可以下降得更快。这时，温度的下降变得与时间成反比 ：$T_0/(1+t)$
- Cauchy分布函数可以用常规的方法进行积分运算

Cauchy分布函数积分运算

$$\int_0^{\Delta w} p(x)dx = \int_0^{\Delta w} \frac{T}{T^2 + x^2}dx$$

$$= T\int_0^{\Delta w} \frac{1}{T^2 + x^2}dx$$

$$= T\left[\frac{1}{T}\operatorname{arctg}\frac{x}{T}\right]_0^{\Delta w}$$

$$= \operatorname{arctg}\frac{\Delta w}{T}$$

Cauchy分布函数积分运算

$$tg(P(\Delta w)) = \frac{\Delta w}{T}$$

Δw=αTtg(P(Δw))

- **Monte Carlo**法：在**(0,1)**中按照均匀分布随机取一数为**P(Δw)**，再取当前的温度，就可以直接地计算出**Δw**
- **Cauchy**训练算法：
  - 将算法**6-2**中的**Boltzmann**分布换成**Cauchy**分布

6.4 相关的几个问题

- **Boltzmann机**
  - 每个神经元可以有一个特殊的阈值，用来限制神经元所获得的激活值

$$net_j = \sum_{k=1}^{n} w_{kj} x_k - \theta_j$$

神经元的状态概率发生变化。**o_j=1**的概率为

$$P_j = \frac{1}{1 + \exp(-\frac{net_j}{T})}$$

Boltzmann机

- **Boltzmann机的目标函数（能量函数）**

$$E = \sum_{k<j} w_{kj} o_k o_j + \sum_{k=1}^{n} o_k \theta_k$$

- **“一致性函数”**

$$E = -\sum_{k<j} w_{kj} o_k o_j - \sum_{k=1}^{n} o_k \theta_k$$

人工热问题

- 特殊热——温度关于能量的变化率
  - 系统在能量跃变边界处的温度叫做临界温度
- 人工特殊热/“伪特殊热”
  - 系统的人工温度关于系统的能量函数（目标函数）的平均变化率
- 临界温度
  - 临界温度时的小量下降，会引起能量函数值的较大变化
  - 系统正处于一个局部极小点附近
- 临界温度点可以通过考察所定义的人工特殊热的变化情况得到

BP算法与Cauchy训练的结合

- Cauchy训练的速度比Boltzmann训练快
- Cauchy训练的速度比BP算法慢
- Cauchy训练有可能使网络逃离局部极小点
- 由BP算法提供直接计算部分，Cauchy算法提供随机部分

$w_{ij}=w_{ij}+\Delta w_{ij}$

$\Delta w_{ij}=\alpha((1-\beta)\delta_j o_i+\beta\Delta w_{ij}')+(1-\alpha)\Delta w_{ij}^{(c)}$

$\alpha\in(0,1)$为学习率, $\beta\in(0,1)$为冲量系数

网络陷入瘫痪

- 执行对网络联接权的压缩
- 如，如果将联接权压缩在（**-a**，**a**）以内，**P. D. Wasserman**曾给出如下建议公式

$$w_{ij} = \frac{2a}{1 + \exp\left(-\frac{w_{ij}}{a}\right)} - a$$

第7章 循环网络

- 主要内容
  - Hopfield网络实现的自相联存储
  - 稳定性分析
  - 统计Hopfield网与Boltzmann机
  - 基本双联存储器(BAM)的结构与训练
  - 几种相联存储网络
  - 用Hopfield网解决TSP问题。

第7章 循环网络

- 重点
  - Hopfield网络实现的自相联存储
  - 基本双联存储器的结构与训练。
- 难点
  - 稳定性分析
  - 用Hopfield网解决TSP问题

第7章 循环网络

7.1 循环网络的组织

7.2 稳定性分析

7.3 统计Hopfield网与Boltzmann机

7.4 双联存储器的结构

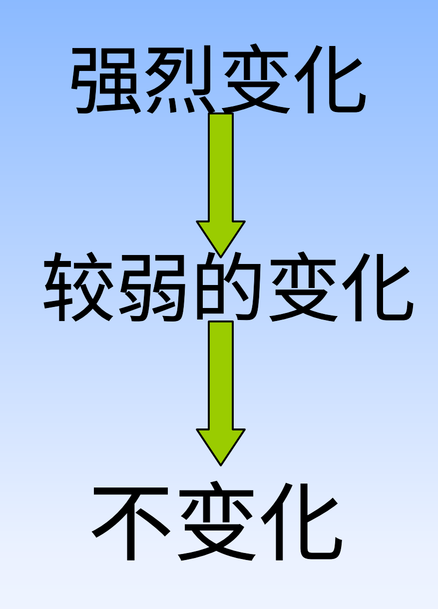
7.5 异相联存储

7.6 其它的双联存储器

7.7 Hopfield网用于解决TSP问题

第7章 循环网络

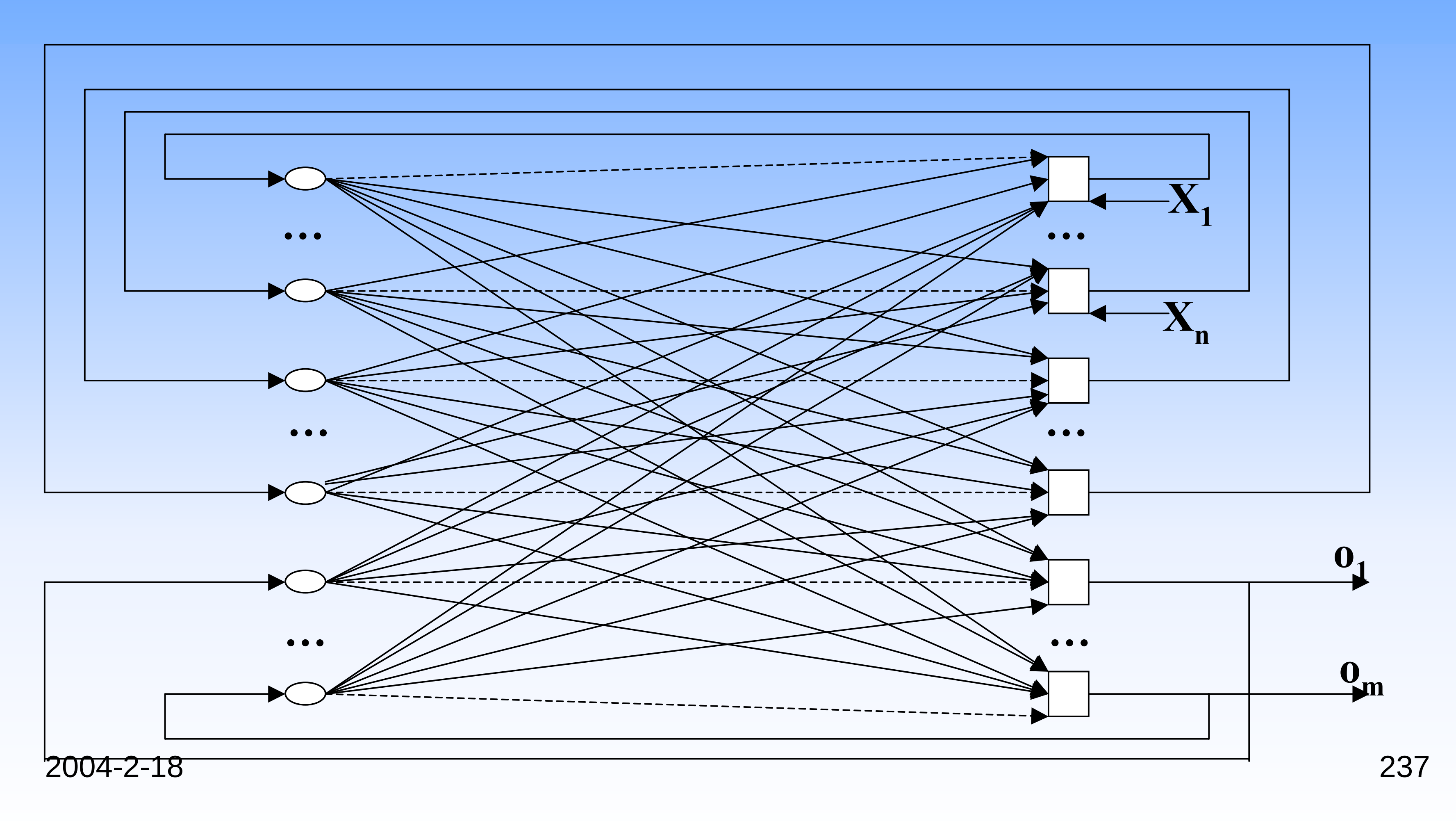
循环网络对输入信号的处理是一个逐渐"修复"、"加强"的过程。



循环网络称为**Hopfield**网

7.1 循环网络的组织

- 网络结构



7.1 循环网络的组织

- 联接：神经元之间都是互联的w_{ij}，每个神经元都没有到自身的联接$w_{ii}=0$。
- 神经元个数h，输入向量维数n，输出向量维数m。h≥n，h≥m，n≥1，m≥1。
- 神经元：输入、输出、隐藏
- 状态变化：非同步、同步
- 输入向量：$X=(x_1, x_2, \dots, x_n)$
- 输出向量：$O=(o_1, o_2, \dots, o_m)$

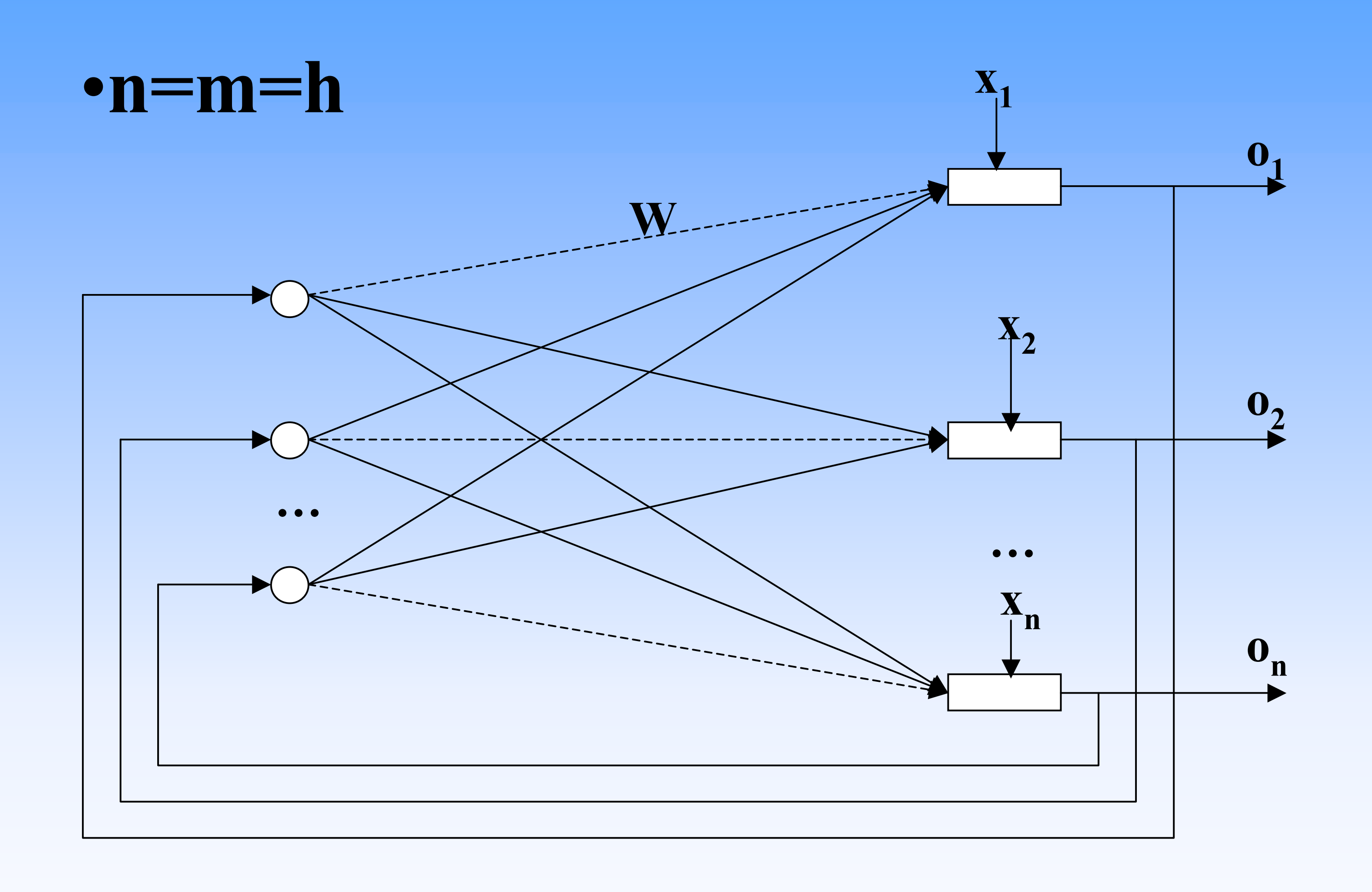
7.1 循环网络的组织

神经元的网络输入： $net_j = \sum_{i=1\&i\neq j}^{n} w_{ij} o_i + x_j$

阈值函数：$o_j = \begin{cases} 1 & \text{if } net_j > \theta_j \\ 0 & \text{if } net_j < \theta_j \\ o_j & \text{if } net_j = \theta_j \end{cases}$

最基本的Hopfield网

- n=m=h



最基本的Hopfield网

- 希望网络的联接矩阵存放的是一组这样的样本，在联想过程中实现对信息的“修复”和“加强”，要求：它的输入向量和输出向量是相同的向量，即，**X=Y**
- 样本集：$S=\{X_1, X_2, \ldots, X_s\}$

最基本的Hopfield网

- 权矩阵：$w_{ij}=\sum_{k=1}^{s} x_{ik} x_{jk}$ 　　　i≠j

  $w_{ii}=0$ 　　　$1 \le i \le n$

- **W**是一个对角线元素为**0**的对称矩阵：
- $\mathbf{W}=\mathbf{X}_1^T \times \mathbf{X}_1+\mathbf{X}_2^T \times \mathbf{X}_2+\ldots+\mathbf{X}_s^T \times \mathbf{X}_s - \mathbf{W}_0$
- **W**是各个样本向量自身的外积的和——网络实现的是自相联映射。

最基本的Hopfield网

- 激活函数：

  改为S形函数后，系统就成为一个连续系统

- 多级循环网络

  除输出向量被反馈到输入层外，其它各层之间的信号传送均执行如下规定：第i-1层神经元的输出经过第i个连接矩阵被送入第i层。

  一般不考虑越层的信号传送、中间的信号反馈和同层的神经元之间进行信号的直接传送

7.2 稳定性分析

- 网络的稳定性是与收敛性不同的问题
- **Cohen**和**Grossberg[1983**年**]:Hopfield**网络的稳定性定理

  如果**Hopfield**网络的联接权矩阵是对角线为**0**的对称矩阵，则它是稳定的
- 用著名的Lyapunov函数作为Hopfield网络的能量函数

Lyapunov函数——能量函数

$$E=-\frac{1}{2}\sum_{i=1}^{h}\sum_{j=1}^{h}w_{ij}o_io_j-\sum_{j=1}^{n}x_jo_j+\sum_{j=1}^{h}\theta_jo_j$$

- 作为网络的稳定性度量
  - $w_{ij}o_io_j$：网络的一致性测度。
  - x_jo_j：神经元的输入和输出的一致性测度。
  - θ_jo_j：神经元自身的稳定性的测度。

当AN_k的状态从o_k变成o_k'

1、AN_k是输入神经元

$$E' = -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_io_j - \sum_{j=1\&j\neq k}^{n} x_jo_j + \sum_{j=1\&j\neq k}^{h} \theta_jo_j -$$

$$-\frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{kj}o'_ko_j - \frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{jk}o_jo'_k - \frac{1}{2}w_{kk}o'_ko'_k -$$

$$-x_ko'_k + \theta_ko'_k$$

$$= -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_io_j - \sum_{j=1\&j\neq k}^{n} x_jo_j + \sum_{j=1\&j\neq k}^{h} \theta_jo_j -$$

$$-\sum_{j=1\&j\neq k}^{h} w_{kj}o'_ko_j - x_ko'_k + \theta_ko'_k$$

当AN$_k$的状态从o$_k$变成o$_k$'

$$\Delta E = E' - E$$

$$= -[\sum_{j=1\&j\neq k}^{h} w_{kj}(o_k' - o_k)o_j] - x_k(o_k' - o_k) + \theta_k(o_k' - o_k)$$

$$= -[\sum_{j=1\&j\neq k}^{h} w_{kj}o_j + x_k - \theta_k](o_k' - o_k) \qquad w_{kk}=0$$

$$= -[\sum_{j=1}^{h} w_{kj}o_j + x_k - \theta_k](o_k' - o_k)$$

$$= -(net_k - \theta_k)\Delta o_k$$

$\Delta E=-(net_k-\theta_k)\Delta o_k$

- AN_k状态的变化：$\Delta o_k=(o_k'-o_k)$
- $\Delta o_k=0$，$\Delta E=0$
- $\Delta o_k>0$，$o_k'=1$& $o_k=0$，o_k由0变到1，

  $net_k>\theta_k$，$net_k-\theta_k>0$

  所以，$-(net_k-\theta_k)\Delta o_k<0$故$\Delta E<0$
- $\Delta o_k<0$, $o_k'=0$& $o_k=1$，o_k由1变到0

  $net_k<\theta_k$，$net_k-\theta_k<0$

  $-(net_k-\theta_k)\Delta o_k<0$故$\Delta E<0$

结论：网络的目标函数总是下降

当AN_k的状态从o_k变成o_k'

2、AN_k不是输入神经元

$$E' = -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_i o_j - \sum_{j=1}^{n} x_j o_j + \sum_{j=1\&j\neq k}^{h} \theta_j o_j -$$

$$-\frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{kj}o_k' o_j - \frac{1}{2}\sum_{j=1\&j\neq k}^{h} w_{jk}o_j o_k' - \frac{1}{2}w_{kk}o_k' o_k' + \theta_k o_k'$$

$$= -\frac{1}{2}\sum_{i=1\&i\neq k}^{h}\sum_{j=1\&j\neq k}^{h} w_{ij}o_i o_j - \sum_{j=1}^{n} x_j o_j + \sum_{j=1\&j\neq k}^{h} \theta_j o_j -$$

$$-\sum_{j=1\&j\neq k}^{h} w_{kj}o_k' o_j + \theta_k o_k'$$

当AN_k的状态从o_k变成o_k'

$$\Delta E = E' - E$$

$$= -\left[\sum_{j=1 \& j \neq k}^{h} w_{kj}(o_k' - o_k)o_j\right] + \theta_k(o_k' - o_k)$$

$$= -\left[\sum_{j=1 \& j \neq k}^{h} w_{kj}o_j - \theta_k\right](o_k' - o_k)$$

$$= -\left[\sum_{j=1}^{h} w_{kj}o_j - \theta_k\right](o_k' - o_k)$$

$$= -(net_k - \theta_k)\Delta o_k$$

无论AN_k的状态是如何变化的，总有$\Delta E \leq 0$

7.3 统计Hopfield网与Boltzmann机

- **统计Hopfield网**
  - 在网络运行中，神经元状态与“人工温度”确定的概率相关
  - 网络运行模拟金属退火过程

$$p_i = \frac{1}{1+\exp(-\frac{net_i - \theta_i}{T})}$$

p_i：AN$_i$的状态取1的概率

net_i：AN$_i$所获网络输入；

θ_i：AN$_i$的阈值；

T：系统的人工温度。

算法 7-1 统计Hopfield网运行算法

1 取一个很大的值作为人工温度T的初值；

2 对网络中每一个神经元AN_i，

2.1 按照相应式子计算相应的概率p_i；

2.2 按照均匀分布，在[0，1]中取一个随机数r；

2.3 如果 $p_i>r$ 则使AN_i的状态为1，

否则使AN_i的状态为0；

3 逐渐降低温度T，如果温度足够低，则算法结束。否则，重复2

Boltzmann机的训练

- **Boltzmann**机是多级循环网络，是**Hopfield**网的一种扩展。
- 神经元AN_i实际输出状态o_i=1的概率为：

$$p_i = \frac{1}{1+\exp(-\frac{net_i - \theta_i}{T})}$$

- T趋近于0时，神经元的状态不再具有随机性，Boltzmann机退化成一般Hopfield网。

Boltzmann机的训练

- Boltzmann机的能量函数(一致性函数)

$$E = -\sum_{i<j} w_{ij} o_i o_j + \sum_{j=1}^{h} \theta_j o_j$$

- 神经元AN_i在运行中状态发生了变化

$$\Delta E_i = E(o_i = 0) - E(o_i = 1) = \sum_j w_{ij} o_j - \theta_i$$

Boltzmann机的训练

- 如果ΔE_i>0，则应该选AN_i输出为1，否则，应该选AN_i输出为0。
- ΔE_i的值越大，神经元AN_i应该处于状态1的概率就应该越大。反之，ΔE_i的值越小，神经元AN_i应该处于状态1的概率就应该越小。从而，o_i=1的概率为：

$$p_i = \frac{1}{1+\exp(-\frac{\Delta E_i}{T})}$$

Boltzmann机的训练

- 处于状态a，b的概率P_a和P_b，对应于$o_i=1$和$o_i=0$，其它的神经元在a，b状态下不变
- $P_a = \gamma p_i$
- $P_b = \gamma (1-p_i)$

$$\frac{P_a}{P_b} = \exp\left(-\frac{E_a - E_b}{T}\right)$$

Boltzmann机的训练

- 网络进行足够多次迭代后，处于某状态的概率与此状态下的能量和此时系统的温度有关。
- 由于高温时网络的各个状态出现的概率基本相同，这就给它逃离局部极小点提供了机会。
- 当系统的温度较低时，如果$E_a<E_b$，则$P_a>P_b$：网络处于较低能量状态的概率较大

Boltzmann机的训练

- **1986年，Hinton和Sejnowski训练方法**
  - 自由概率P_{ij}^{-}：没有输入时AN_i和AN_j同时处于激发状态的概率。
  - 约束概率P_{ij}^{+}：加上输入后AN_i和AN_j同时处于激发状态的概率。
  - 联接权修改量：$\Delta w_{ij} = \alpha (P_{ij}^{+} - P_{ij}^{-})$

算法7-2 Boltzmann机训练算法

1 计算约束概率

1.1 对样本集中每个样本，执行如下操作：

1.1.1 将样本加在网络上（输入向量及其对应的输出向量）；

1.1.2 让网络寻找平衡；

1.1.3 记录下所有神经元的状态；

1.2 计算对所有的样本，AN_i和AN_j的状态同时为1的概率P_{ij}^{+}；

算法7-2 Boltzmann机训练算法

2 计算自由概率

2.1 从一个随机状态开始，不加输入、输出，让网络自由运行，并且在运行过程中多次纪录网络的状态；

2.2 对所有的AN_i和AN_j，计算它们的状态同时为**1**的概率P_{ij}^-；

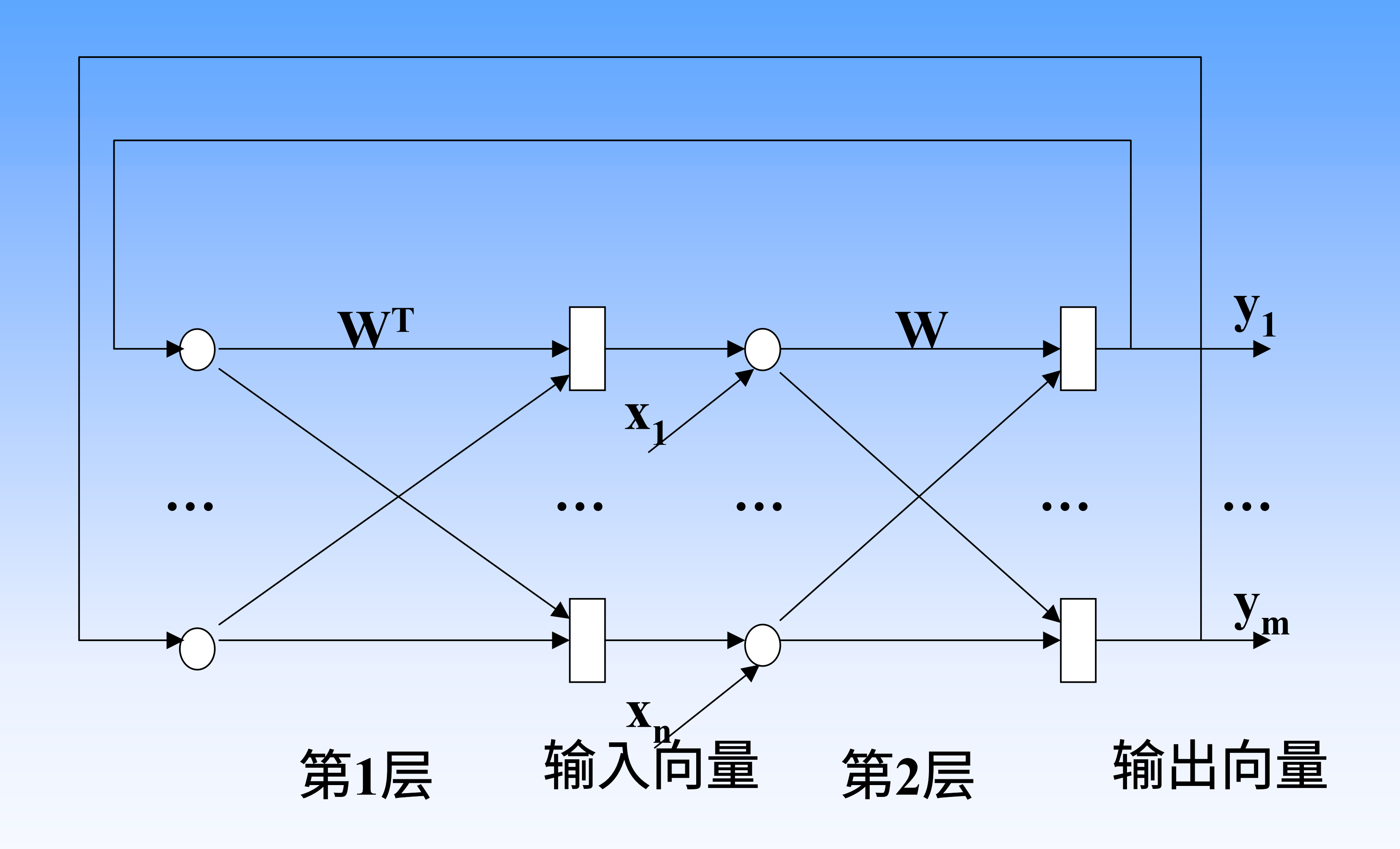
3 对权矩阵进行调整

$$\Delta w_{ij} = \alpha (P_{ij}^+ - P_{ij}^-)$$

7.4 双联存储器的结构

- 智力链
  - 从一件事想到另一件事，“唤回失去的记忆”。
- 自相联
- 异相联
  - 双联存储器（**Bidirectional Associative Memory—BAM**）。
- 双联存储器具有一定的推广能力
  - 它对含有一定缺陷的输入向量，通过对信号的不断变换、修补，最后给出一个正确的输出。

基本的双联存储器结构



网络运行

Y=F(XW)

$X=F(YW^T)$

$X=(x_1, x_2, \ldots, x_n)$

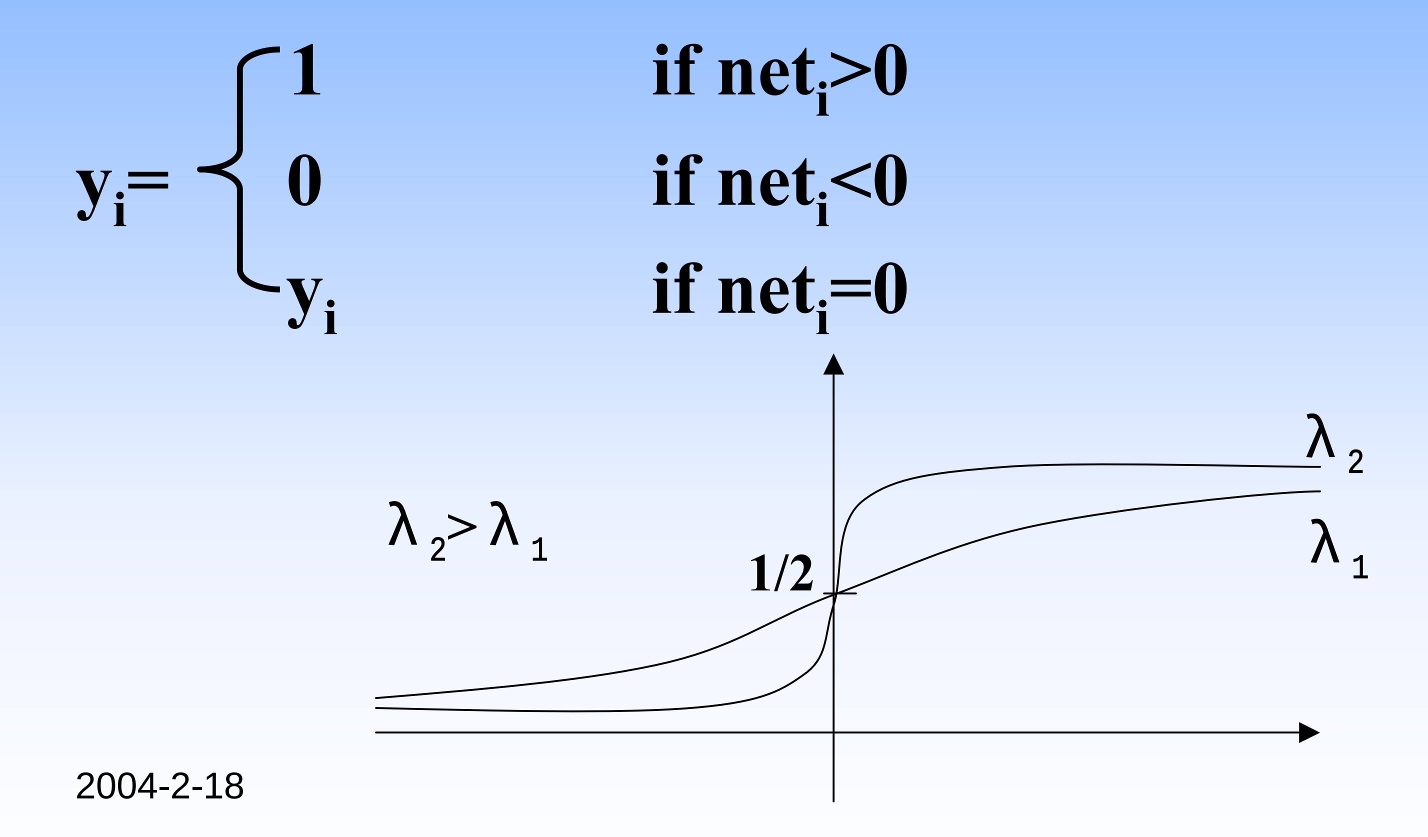
$Y=(y_1, y_2, \ldots, y_m)$

F为神经元的激活函数，一般可采用S形函数

$$y_i = \frac{1}{1+\exp(-\lambda net_i)}$$

激活函数——阈值函数

- 随着λ的增加，该函数趋近于阈值为0的阈值函数。



基本BAM的稳定

- **Kosko(1987)**：
  - 基本的双联存储器无条件稳定——联接权矩阵是互为转置矩阵。
- 当输入向量的维数与输出向量的维数相同时，**W**为方阵，此时如果联接矩阵**W**是对称的，则基本的双联存储器退化成一个**Hopfield**网

7.5 异相联存储

- 样本集：$S=\{(X_1,Y_1),(X_2,Y_2)\ldots,(X_s,Y_s)\}$
- 权矩阵

$$W=\sum_{i=1}^{s} X_i^T \times Y_i$$

- 网络需要对输入向量进行循环处理的情况
  - 当输入向量中含有“噪音”
  - 样本集所含的信息超出网络的容量

容量

- **Kosko（1987），一般情况下，相联存储器的容量不会超过网络最小层神经元的个数min**
- **Haines和Hecht-Nielson（1988），"非均匀"网络的容量最多可以达到2^{min}**
- **R. J. McEliece、E. C. Posner、E. R. Rodemich**
  - 用户随机地选择L个状态
  - 每个向量中有$4+\log_2 min$个分量为1，其它为-1
  - 98%的向量成为稳定状态

$$L < \frac{0.68 \min^2}{(\log_2 \min + 4)^2}$$

7.6 其它的双联存储器

- 具有竞争的双联存储器
  - 可通过附加侧联接实现竞争。这些权构成另一个主对角线元素为正值，其它元素为负值的权矩阵。
  - Cohen-Grossberg定理指出，如果权矩阵是对称的，则网络是稳定。
  - 即使权矩阵不对称，网络通常也是稳定的。但是目前还不知道哪一类权矩阵会引起不稳定

7.6 其它的双联存储器

- 连续的双联存储器
  - **Kosko**（**1987**）证明，神经元的状态非同步变换，而且这些神经元使用其他激励函数，仍然是稳定的，且有更强的表达能力
- 自适应双联存储器
  - 最简单的方法是使用**Hebb**学习律进行训练。
  - $\Delta \mathbf{w}_{ij} = \alpha\, \mathbf{o}_i \mathbf{o}_j$

7.7 Hopfield网解决TSP问题

- **1985**年，**J. J. Hopfield**和**D. W. Tank**用循环网求解**TSP**。试验表明，当城市的个数不超过**30**时，多可以给出最优解的近似解。而当城市的个数超过**30**时，最终的结果就不太理想了
- n个城市间存在n!/(2n)条可能路径
- 设问题中含有**n**个城市,用**n*n**个神经元构成网络

7.7 Hopfield网解决TSP问题

- d_{xy}——城市X与城市Y之间的距离；
- y_{xi}——城市X的第i个神经元的状态：

$$y_{xi}=\begin{cases}1 & \text{城市X在第i个被访问} \\ 0 & \text{城市X不在第i个被访问}\end{cases}$$

- $w_{xi,yj}$——城市X的第i个神经元到城市Y的第j个神经元的连接权。

7.7 Hopfield网用于解决TSP问题

例如：四个城市X、Y、Z、W

城市名	访问顺序标示			
	1	2	3	4
X	0	1	0	0
Y	0	0	0	1
Z	1	0	0	0
W	0	0	1	0

7.7 Hopfield网用于解决TSP问题

- 联接矩阵

$$w_{xi,yj} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - \zeta d_{xy}(\delta_{ji+1} + \delta_{ji-1})$$

$$\delta_{ij} = \begin{cases} 1 & \text{如果} i = j \\ 0 & \text{如果} i \neq j \end{cases}$$

网络的能量函数

$$
E = \frac{A}{2}\sum_{x}\sum_{i}\sum_{j \neq i} y_{xi} y_{xj}
$$

$$
+ \frac{B}{2}\sum_{i}\sum_{x}\sum_{x \neq z} y_{xi} y_{zi} + \frac{C}{2}\left(\sum_{x}\sum_{i} y_{xi} - n\right)^2
$$

$$
+ \frac{D}{2}\sum_{x}\sum_{z \neq x}\sum_{i} d_{xz} y_{xi}\left(y_{zi+1} + y_{zi-1}\right)
$$

网络的能量函数

- A、B、C、D为惩罚因子

第1项 $$\frac{A}{2}\sum_{x}\sum_{i}\sum_{j\neq i} y_{xi} y_{xj}$$

- 仅当所有的城市最多只被访问一次时取得极小值0。

网络的能量函数

第2项 $$+\frac{B}{2}\sum_{i}\sum_{x}\sum_{x\neq z}y_{xi}y_{zi}$$

- 仅当每次最多只访问一个城市时取得极小值0。

网络的能量函数

第3项
$$+\frac{C}{2}\left(\sum_{x}\sum_{i}y_{xi}-n\right)^2$$

- 当且仅当所有的n个城市一共被访问n次时才取得最小值0。

网络的能量函数

第4项 $$+\frac{D}{2}\sum_{x}\sum_{z\neq x}\sum_{i} d_{xz} y_{xi}\left(y_{zi+1}+y_{zi-1}\right)$$

- 表示按照当前的访问路线的安排，所需要走的路径的总长度

第8章 自适应共振理论

主要内容

1. **ART**模型的总体结构
2. 各模块功能
3. 比较层
4. 识别层联接矩阵的初始化
5. 识别过程与比较过程
6. 查找的实现
7. **ART**的训练。

第8章 自适应共振理论

- 重点
  - ART模型的总体结构
  - 各模块功能
  - 识别过程与比较过程
  - 查找的实现。
- 难点
  - 比较层与识别层联接矩阵的初始化

第8章 自适应共振理论

8.1 ART的结构

8.2 ART的初始化

8.2.1 T的初始化

8.2.2 B的初始化

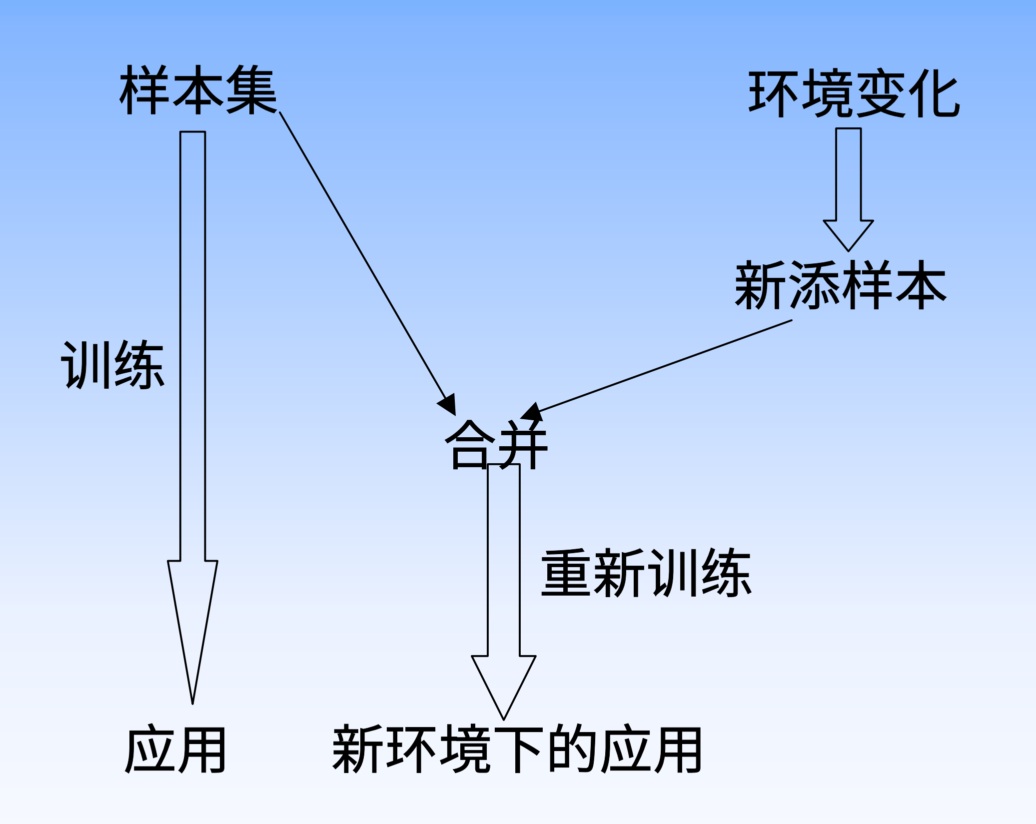
8.2.3 ρ的初始化

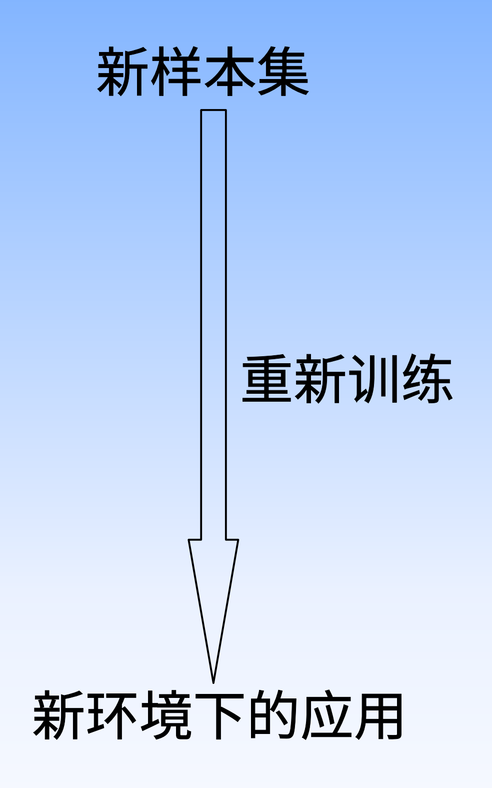
8.3 ART的实现

识别、比较 、查找 、训练

第8章 自适应共振理论

网络与样本





第8章 自适应共振理论

- Carpenter和Grossberg在1986年：4个样本组成样本集。这4个样本被周期性地提交给网络，网络难以收敛
- 问题分析

1. 网络的LTM只是它最后获得训练时系统所面临的样本集所蕴含的内容
2. 进行的是全部修改，而非部分修改
3. 最好是能够实现相关部分的修改

第8章 自适应共振理论

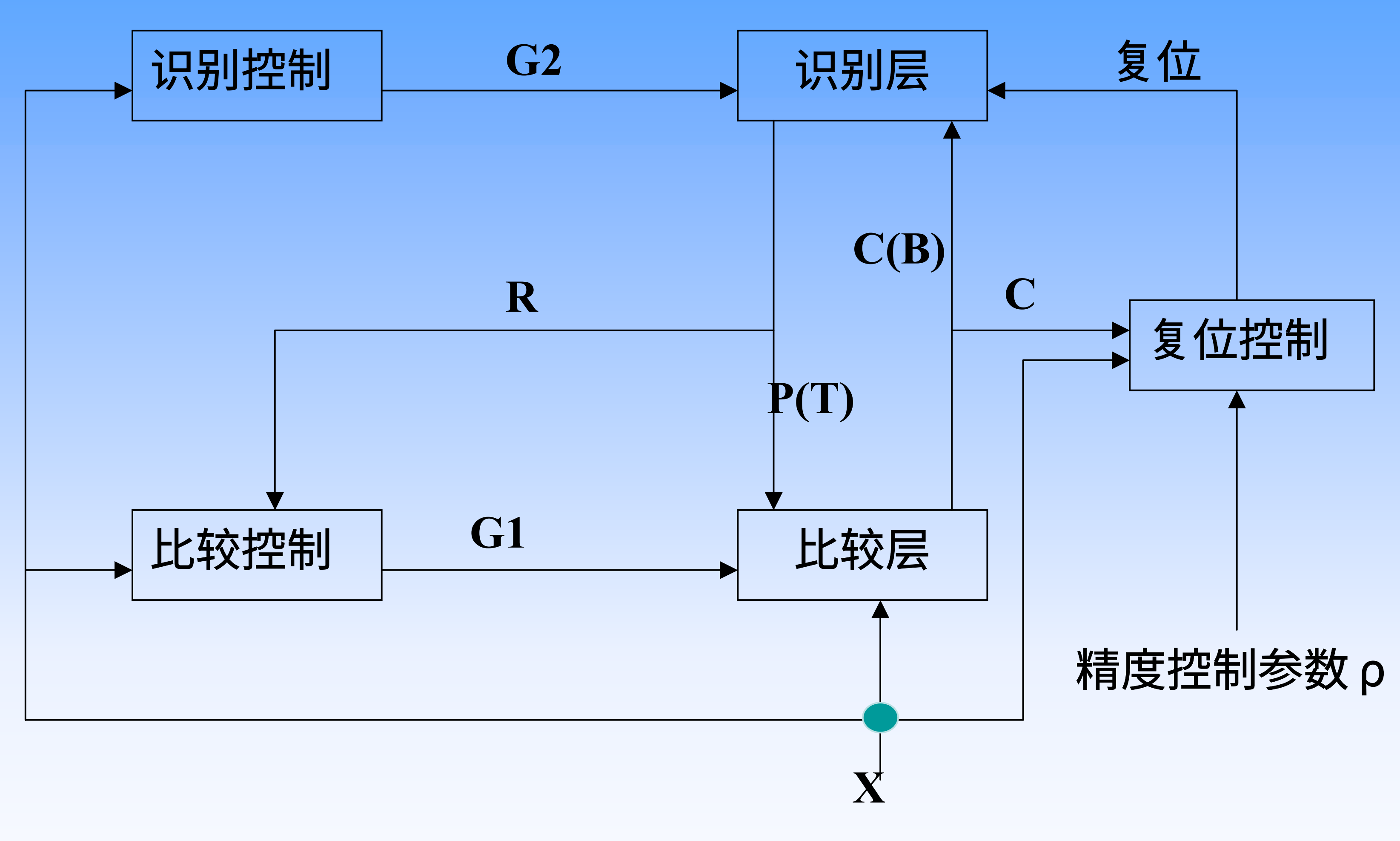
- 网络的可塑性需要的4项功能
  - 样本的分类功能
  - 分类的识别功能
  - 比较功能
  - 类的建立功能
- Grossberg等：自适应共振理论——ART (Adaptive Resonance Theory)
- ART1、ART2。

8.1 ART的结构

- 稳定性与可塑性是不同的
  - **ART**是一种循环网络
- 保证可塑性的操作要求分析

新输入向量与现存模式 { 相似：修改相匹配的模式 } 不匹配的现存模式不被修改
新输入向量与现存模式 { 不相似：建立一个新模式 } 不匹配的现存模式不被修改

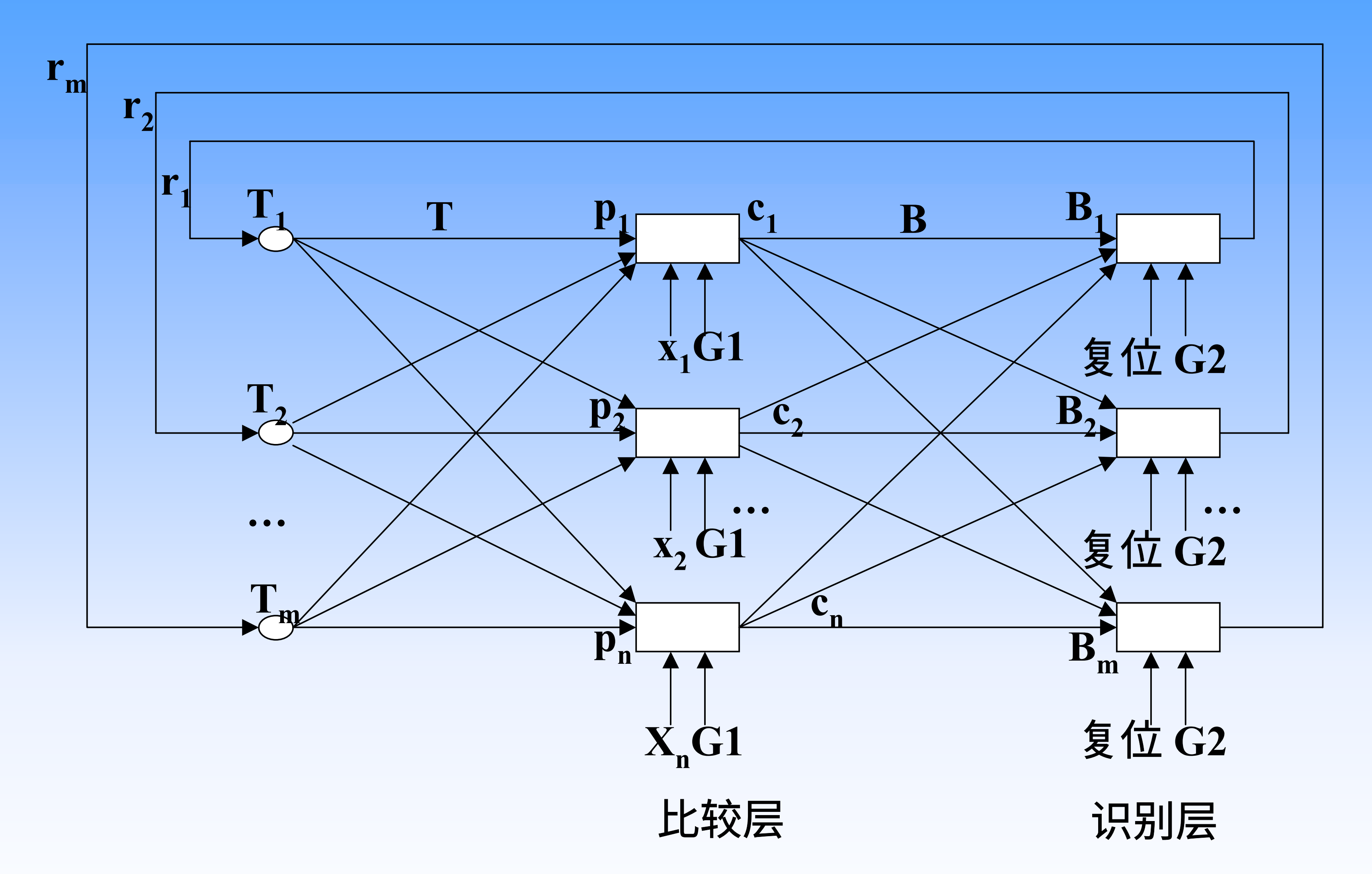
ART总体结构图



B—Bottom-up

T—Top-down

以比较层和识别层为中心讨论5个功能模块



8.1 ART的结构

$X=(x_1, x_2, \ldots, x_n)$

$R=(r_1, r_2, \ldots, r_m)$

$C=(c_1, c_2, \ldots, c_n)$

$P=(p_1, p_2, \ldots, p_n)$

$T_i=(t_{i1}, t_{i2}, \ldots, t_{in}) \quad 1 \leq i \leq m$

$B_i=(b_{1i}, b_{2i}, \ldots, b_{ni}) \quad 1 \leq i \leq m$

8.1 ART的结构

- t_{ij}表示识别层的第i个神经元到比较层的第j个神经元的联接权
- b_{ij}表示比较层的第i个神经元到识别层的第j个神经元的联接权
- p_i为比较层的第i个神经元的网络输入

$$p_i = \sum_{j=1}^{m} r_j t_{ji}$$

比较层输出信号控制

$$G1 = \neg(r_1 \vee r_2 \vee \ldots \vee r_m) \wedge (x_1 \vee x_2 \vee \ldots \vee x_n)$$

识别层输出信号控制

$$G2 = x_1 \vee x_2 \vee \ldots \vee x_n$$

比较层

- 执行二-三规则

if $x_i+p_i+G1 \geq 2$ then $c_i=1$

if $x_i+p_i+G1 < 2$ then $c_i=0$

- 待命期 (X=0)
- 工作期 (X≠0)

$$p_i = \sum_{j=1}^{m} r_j t_{ji}$$

$C=X$

$$= r_k t_{ki}$$

$P=T_k$

$r_k=1$

$$= t_{ki}$$

$c_i=x_i \wedge p_i$

识别层

- 识别层实现竞争机制
- B_k与C有最大的点积

$$\sum_{i=1}^{n} b_{ik} c_i = \max\{\sum_{i=1}^{n} b_{ij} c_i \mid 1 \le j \le m\}$$

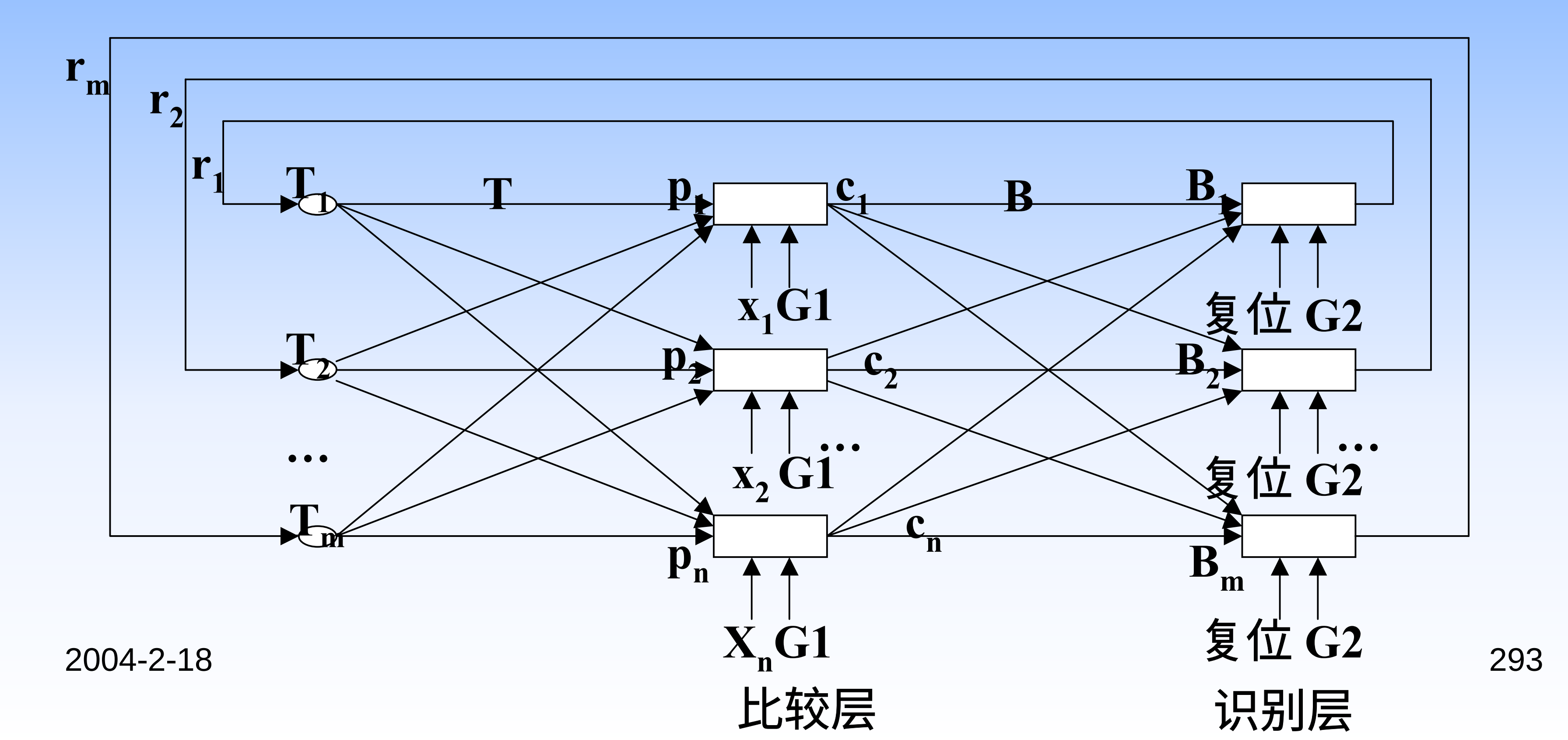
- X的“暂定”代表RN_k所获得的网络输入为

$$\sum_{i=1}^{n} b_{ik} c_i$$

与RN_1，RN_2，…，RN_m相对应
向量B_1，B_2，…，B_m代表不同分类

识别层

- 与RN_1，RN_2，…，RN_m相对应向量B_1，B_2，…，B_m代表不同分类
- T_1，T_2，…，T_m与向量B_1，B_2，…，B_m相对应——类的不同表示形式



系统复位控制

X与C的相似度

$$s = \frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} x_i}$$

- $s \geq \rho$，当前处于激发态的RN_k所对应的B_k、T_k为X的类表示；
- $s < \rho$，此RN_k所对应的B_k、T_k不能很好地代表X，需要重新寻找

8.2 ART的初始化

- **T**的初始化
  - 矩阵**T**的所有元素全为**1**
- **B**的初始化

  $b_{ij}<L/(L-1+n)$

  - **n**为输入向量的维数；**L**为一个大于**1**的常数，其值应该与输入向量的维数相关
  - T_k、B_k是RN_k对应类的两种不同表示
- ρ 的初始化
  - $\rho \in [0,1]$

8.3 ART的实现

- 四个阶段：识别、比较、查找、训练

一、识别

- X 未被加在网上时(X=0)
  - $G2 = x_1 \lor x_2 \lor \ldots \lor x_n)=0$
  - $R=(r_1, r_2, \ldots, r_m)=(0, 0, \ldots, 0)$
- X被加在网络上时(X≠0)
  - $G1 = \neg (r_1 \lor r_2 \lor \ldots \lor r_m) \land (x_1 \lor x_2 \lor \ldots \lor x_n) = G2 = 1$
  - R=0导致$P=(p_1, p_2, \ldots, p_m)= (0, 0, \ldots, 0)$

8.3 ART的实现

- 在识别层，每个$\mathbf{RN_k}$完成的操作
  - 计算$\sum b_{ik}c_i$
  - 接收来自其它**RN**的抑制信号，并向其它的**RN**发出抑制信号
  - 确定自己的输出状态
  - 完成输出
- RN之间的抑制连接与抑制信号
- 如果RN_k输出1，则表明，在本轮识别中，X暂时被认为是属于该RN_k所对应的类

二、 比较

- X归于RN_k，RN_k的输出值1被分别以权重t_{kj}传送到比较层
- 向量P就是向量T_k
- T的初始化及训练保证了T的每个元素取值为0或者1
- B_k与T_k根据RN_k进行对应，互为变换形式
- 如果对于所有的j，$1 \leq j \leq n$，$p_j=x_j$，则表示X获得良好的匹配。如果存在j，使得$p_j \neq x_j$，则表明X与相应的“类”的代表向量并不完全一致

二、 比较

- 当系统复位控制模块计算**X**和**C**的相似度**s**
- 如果**s**≥ρ，表明本轮所给出的类满足精度要求。查找成功，系统进入训练周期
- 如果**s**<ρ，表明本轮所给类不满足精度要求。
  - 复位模块要求识别层复位，使所有**RN**输出**0**
  - 系统回到开始处理**X**的初态，重新进行搜索
  - 复位信号屏蔽本次被激发的**RN**，在下一轮匹配中，该**RN**被排除在外，以便系统能够找到其它更恰当的**RN**

三、 查找

- 如果s≥ρ，认为网络查找成功，此时分类完成，无需再查找
- 如果s<ρ，表明本轮实现的匹配不能满足要求，此时需要寻找新的匹配向量
- 查找过程

三、 查找

1 复位模块向识别层发出复位信号

2 所有RN被抑制：$R=(r_1, r_2, \ldots, r_m)=(0, 0, \ldots, 0)$，上轮被激发的RN被屏蔽

3 G1的值恢复为1

4 X的值再次被从比较层送到识别层：C=X

5 不同的RN被激发，使得不同的$P(T_k)$被反馈到比较层

6 比较层进行相应的比较，并判定本次匹配是否满足要求

三、 查找

7 如果本次匹配不成功，则重复1∽6直到如下情况之一发生

7.1 本轮匹配成功。表明已找到一个与**X**匹配较好的模式，此时，网络进入训练期，对这个匹配的模式进行适当的修改，使它能更好地表示**X**

7.2 网络中现存的模式均不匹配。因此，网络需要重新构造一个新模式表达此类

三、 查找

- 网络用一个还未与任何类关联的RN来对应X所在的类
  - 根据X修改与此RN对应的T_k、B_k
  - 被网络选中的RN_k对应的T_k=（1，1，…，1）
  - P=（1，1，…，1）被送入比较层。
  - C=X∧P=X，被送入系统复位控制模块，s=1。而ρ≤1，所以，s≥ρ。匹配获得成功
  - 网络进入训练期

三、 查找

- 首先被选中的RN不一定对应X属于的类
  - 受B取法的影响，有时候，获得最大激励值的RN对应的类不一定是X所属的类
- 例如：设n=5，三个输入向量为：

X_1=（1，0，0，0，0）

X_2=（1，0，0，1，1）

X_3=（1，0，0，1，0）

三、 查找

$$b_{ik} = \frac{Lc_i}{L - 1 + \sum_{j=1}^{n} c_j}$$

- 假定用(2/2-1+5)初始化B，当X_1、X_2被输入时，RN_1、RN_2分别被激发
- T_1、T_2、B_1、B_2分别取如下值
  - T_1=(1,0,0,0,0)，B_1=(1,0,0,0,0)
  - T_2=(1,0,0,1,1)，B_2=(0.5,0,0,0.5,0.5)
- 当X_3被输入系统时，RN_1、RN_2获得的激励值都是1
  - RN_2被选中 ,则成功

三、 查找

- RN_1被选中，则不满足精度要求，网络就需要进入查找工作阶段
- 具体过程

1、 RN_1获胜

2、C取值（1，0，0，0，0）

3、 $$s = \sum_{i=1}^{5} c_i / \sum_{i=1}^{5} x_i = 0.5$$

三、 查找

4、s＜ρ

5、RN_1被屏蔽

6、网络进入第二个查找周期，RN_2获胜

7、C取值（1，0，0，1，0）

8、 $$s = \sum_{i=1}^{5} c_i / \sum_{i=1}^{5} x_i = 1.0$$

9、满足精度要求，停止查找，进入训练期

三、　查找

- L是常数，当L取其它的值时，将会有不同的结果
- 当RN被系统认为是不能满足精度要求后，在继续查找过程中，一直被屏蔽
- “查找周期”：网络的五个功能模块之间互相影响，加上信号的反馈，使得网络中的信号较为复杂

四、 训练

- **T_k、B_k的修改**

$$b_{ik} = \frac{Lc_i}{L - 1 + \sum_{j=1}^{n} c_j}$$

$$t_{ki} = c_i$$

四、 训练

- T的元素只可能从1变成0，不可能从0变成1：用1初始化T的所有元素
- 如果RN_k对应的模式代表类{X_1，X_2，…，X_d}，则有$T_k = X_1 \wedge X_2 \wedge \ldots \wedge X_d$
- 网络将向量共有的东西作为它的类表示，符合一般意义下“共同特征”的要求

四、 训练

$$b_{ik} = \frac{Lc_i}{L - 1 + \sum_{j=1}^{n} c_j}$$

中含有重要因子

$$\sum_{j=1}^{n} c_j$$

四、 训练

- Σc_j可以看成向量C的一个度量
  - 越大，产生的权值就越小；
  - 越小，产生的权值就越大。
  - 当一个向量是另一个向量的子集时，能够获得较好的操作
- 例如——假设无Σc_j

  X_1=（1，0，0，0，0）

  X_2=（1，0，0，1，1）

  X_3=（1，0，0，1，0）

四、 训练

- 设X_1、X_2分别使RN_1、RN_2激发
- 设$T_1 = X_1$、$T_2 = X_2$
- 相应式子中没有$\sum c_j$，则$B_1 = T_1$、$B_2 = T_2$
- 当X_1再一次被输入时，RN_1、RN_2因为获得的网络输入相同而都有被选中的可能
- 如果RN_2被选中，则会导致网络运行错误，使得原有的分类被严重破坏

四、 训练

•具体过程

① X_1被再次输入，导致RN_2被选中；

② 识别层将T_2送入比较层：$P= T_2$；

③ 此时，$C=P\wedge X_1=X_1$；

④ 复位控制模块根据C与X_1计算出$s=1$；

⑤ 因为$s>\rho$，所以对网络进行训练：$T_2=C$。原值被破坏了。当选择一个适当的L，同时在调整B时保留Σc_j，就可以避免这个问题

四、 训练

- 网络的分类并不是一成不变的
- 继续使用上面例子中的输入向量，取L=6，初始化使B的所有元素均取值0.6

1、 X_1的输入导致RN_1被激发；B_1被训练后取值为（1，0，0，0，0）

2、输入X_2时，RN_1、RN_2所获得的网络输入分别为1和1.8，这导致RN_2被激发；B_2被训练后取值为（0.6，0，0，0.6，0.6）

四、 训练

3、如果X_1再次被输入，RN_1、RN_2所获得的网络输入分别为1和0.6，从而正确的神经元被激发；如果X_2再次被输入，RN_1、RN_2所获得的网络输入分别为1和1.8，从而也仍然有正确的神经元被激发

4、当X_3被输入时，RN_1、RN_2所获网络输入分别为1和1.2，从而RN_2被激发，此时，T_2=（1，0，0，1，1）被送入比较层，使得$C=T_2 \wedge X_3=X_3$。从而导致$s=1>\rho$

四、 训练

5、网络进入训练：T_2、B_2被修改

T_2=（1，0，0，1，0）

B_2=（6/7，0，0，6/7，0）

6、当再次输入X_2时，RN_1、RN_2所获得的网络输入分别为：1和12/7，这再次导致RN_2被激发。但是，此时识别层送给比较层的T_2=（1，0，0，1，0）。从而有s=2/3，如果系统的复位控制参数ρ>2/3，此时系统会重新为X_3选择一个新的神经元

四、 训练

- 可以让ART在训练完成后，再投入运行